

Higher Dimensional Modal Logic

Cristian Prisacariu

Dept. of Informatics, University of Oslo, – P.O. Box 1080 Blindern, N-0316 Oslo, Norway.

cristi@ifi.uio.no

Higher dimensional automata (*HDA*) are a model of concurrency that can express most of the traditional partial order models like Mazurkiewicz traces, pomsets, event structures, or Petri nets. Modal logics, interpreted over Kripke structures, are the logics for reasoning about sequential behavior and interleaved concurrency. Modal logic is a well behaved subset of first-order logic; many variants of modal logic are decidable. However, there are no modal-like logics for the more expressive *HDA* models. In this paper we introduce and investigate a modal logic over *HDAs* which incorporates two modalities for reasoning about “during” and “after”. We prove that this general higher dimensional modal logic (*HDML*) is decidable and we define an axiomatic system for it. We also show how, when the *HDA* model is restricted to Kripke structures, a syntactic restriction of *HDML* becomes the standard modal logic. Then we isolate the class of *HDAs* that encode Mazurkiewicz traces and show how *HDML*, with natural definitions of corresponding *Until* operators, can be restricted to LTrL (the linear time temporal logic over Mazurkiewicz traces) or the branching time ISTL. We also study the expressiveness of the basic *HDML* language wrt. bisimulations and conclude that *HDML* captures the split-bisimulation.

Contents

1	Introduction	2
2	Modal Logic over Higher Dimensional Automata	3
2.1	Decidability of <i>HDML</i>	5
2.2	Axiomatic system for <i>HDML</i>	9
3	Examples of Encodings into Higher Dimensional Modal Logic	14
3.1	Encoding standard modal logic into <i>HDML</i>	15
3.2	Adding an Until operator and encoding standard temporal logic	16
3.3	Partial order models and their logics in <i>HDML</i>	17
4	Expressiveness in terms of bisimulations	21
5	Conclusion	23
A	Completeness	26

1 Introduction

This paper extends [1] by adding all the proofs and some more explanations. Moreover, it corrects some essential errors that appeared in the proofs of soundness and completeness of the axiomatic system of [1]. The present paper also adds new results that stem from two comments that this work attracted. We discuss the expressive power of the basic logic wrt. bisimulations, concluding that it captures the split-bisimulation. We investigate more carefully the extension of the basic language with the *Until* operator; we define precisely two kinds of *Until*, and we use the LTL-like to encode the LTrL logic and the CTL-like to encode the ISTL logic.

Higher dimensional automata (HDAs) are a general formalism for modeling concurrent systems [2, 3]. In this formalism concurrent systems can be modeled at different levels of abstraction, not only as all possible interleavings of their concurrent actions. *HDAs* can model concurrent systems at any granularity level and make no assumptions about the durations of the actions, i.e., refinement of actions [4] is well accommodated by *HDAs*. Moreover, *HDAs* are not constrained to only before-after modeling and expose explicitly the choices in the system. It is a known issue in concurrency models that the combination of causality, concurrency, and choice is difficult; in this respect, *HDAs* and Chu spaces [5] do a fairly good job [6].

Higher dimensional automata are more expressive than most of the models based on partial orders or on interleavings (e.g., Petri nets and the related Mazurkiewicz traces, or the more general partial order models like pomsets or event structures). Therefore, one only needs to find the right class of *HDAs* in order to get the desired models of concurrency.

Work has been done on defining temporal logics over Mazurkiewicz traces [7] and strong results like decidability and expressive completeness are known [8, 9]. For more general partial orders some temporal logics become undecidable [10]. For the more expressive event structures there are fewer works; a modal logic is investigated in [11].

There is hardly any work on logics for higher dimensional automata [6] and, as far as we know, there is no work on *modal logics for HDAs*. In practice, one is more comfortable with modal logics, like temporal logics or dynamic logics, because these are generally decidable (as opposed to full first-order logic, which is undecidable).

That is why in this paper we introduce and develop a logic in the style of standard modal logic. This logic has *HDAs* as models, hence, the name *higher dimensional modal logic (HDML)*. This is our basic language to talk about general models of concurrent systems. For this basic logic we prove decidability using a form of filtration argument, and we show how compactness fails. Also, we provide an axiomatic system and prove it is sound and complete for the higher dimensional automata. *HDML* in its basic variant is shown to become standard modal logic when the language and the higher dimensional models are restricted in a certain way.

HDML contrasts with standard temporal/modal logics in the fact that *HDML* can reason about *what holds “during” some concurrent events are executing*. The close related logic for distributed transition systems of [12] is in the same style of reasoning only about what holds “after” some concurrent events have finished executing. As we show in the examples section, the “after” logics can be encoded in *HDML*, hence also the logic of [12].

The other purpose of this work is to provide a general framework for reasoning about concurrent systems at any level of abstraction and granularity, accounting also for choices and independence of actions. Thus, the purpose of the examples in Section 3 is to show that studying *HDML*, and particular variants of it, is fruitful for analyzing concurrent systems and their logics. In this respect we study variants of higher dimensional modal logic inspired by temporal logic and dynamic logic. Already in Section 3.2 we add to the basic language two kinds of *Until* operator, in the style of linear and branching time temporal logics. We show how this variant of *HDML*, when interpreted over the class of *HDAs* corresponding to Kripke structures, can be particularized just by syntactic restrictions to CTL [13]. A second variant, in Section 3.3, decorates the *HDML* modalities with labels. This multi-modal variant of *HDML* together with the LTL-like *Until* operator, when interpreted over the class of *HDAs* that encodes Mazurkiewicz traces, becomes LTrL [9] (the linear time temporal logic over Mazurkiewicz traces).

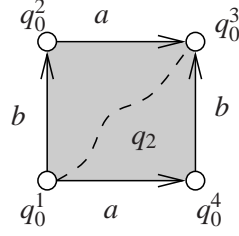


Figure 1: Example of a *HDA* with two concurrent events labeled by *a* and *b*.

2 Modal Logic over Higher Dimensional Automata

In this section we define a higher dimensional automaton (*HDA*) following the definition and terminology of [3, 6]. Afterwards we propose *higher dimensional modal logic (HDML)* for reasoning about concurrent systems modeled as *HDAs*. The semantic interpretation of the language is defined in terms of *HDAs* (i.e., the *HDAs*, with a valuation function attached, are the models we propose for *HDML*).

For an intuitive understanding of the *HDA* model consider the standard example [6, 3] pictured in Figure 1. It represents a *HDA* that models two concurrent events which are labeled by *a* and *b* (one might have the same label *a* for both events). The *HDA* has four states, q_0^1 to q_0^4 , and four transitions between them. This would be the standard picture for interleaving, but in the case of *HDA* there is also a square q_2 . Traversing through the interior of the square means that both events are executing. When traversing on the lower transition means that event one is executing but event two has not started yet, whereas, when traversing through the upper transition it means that event one is executing and event two has finished already. In the states there is no event executing, in particular, in state q_0^3 both events have finished, whereas in state q_0^1 no event has started yet.

In the same manner, *HDAs* allow to represent three concurrent events through a cube, or more events through hypercubes. Causality of events is modeled by sticking such hypercubes one after the other. For our example, if we omit the interior of the square (i.e., the grey q_2 is removed) we are left with a description of a system where there is the choice between two sequences of two events, i.e., $a; b + b; a$.

Definition 2.1 (higher dimensional automata). A cubical set $H = (Q, \bar{s}, \bar{t})$ is formed of a family of sets $Q = \bigcup_{n=0}^{\infty} Q_n$ with all sets Q_n disjoint, and for each n , a family of maps $s_i, t_i : Q_n \rightarrow Q_{n-1}$ with $1 \leq i \leq n$ which respect the following cubical laws:

$$\alpha_i \circ \beta_j = \beta_{j-1} \circ \alpha_i, \quad 1 \leq i < j \leq n \text{ and } \alpha, \beta \in \{s, t\}. \quad (1)$$

In H , the \bar{s} and \bar{t} denote the collection of all the maps from all the families (i.e., for all n). A higher dimensional structure (Q, \bar{s}, \bar{t}, l) over an alphabet Σ is a cubical set together with a labeling function $l : Q_1 \rightarrow \Sigma$ which respects $l(s_i(q)) = l(t_i(q))$ for all $q \in Q_2$ and $i \in \{1, 2\}$.¹ A higher dimensional automaton $(Q, \bar{s}, \bar{t}, l, I, F)$ is a higher dimensional structure with two designated sets of initial and final cells $I \subseteq Q_0$ and $F \subseteq Q_0$.

We call the elements of Q_0, Q_1, Q_2, Q_3 respectively *states*, *transitions*, *squares*, and *cubes*, whereas the general elements of Q_n are called *n-dimensional cubes* (or *hypercubes*). We call generically an element of Q a *cell* (also known as *n-cell*). For a transition $q \in Q_1$ the $s_1(q)$ and $t_1(q)$ represent respectively its source and its target cells (which are *states* from Q_0 in this case). Similarly for a general cell $q \in Q_n$ there are n source cells and n target cells all of dimension $n - 1$. Intuitively, an *n-dimensional cell* q represents a snapshot of a concurrent system in which n events are performed at the same time, i.e., concurrently. A source cell $s_i(q)$ represents the snapshot of the system before the starting of the i^{th} event, whereas the target cell $t_i(q)$ represents the snapshot of the system immediately after the termination of the i^{th} event. A transition of Q_1 represents a snapshot of the system in which a single event is performed.

¹Later, in Definition 3.13, the labeling is extended naturally to all cells.

The cubical laws account for the geometry (concurrency) of the *HDA*s; there are four kinds of cubical laws depending on the instantiation of α and β . For the example of Figure 1 consider the cubical law where α is instantiated to t and β to s , and $i = 1$ and $j = 2$: $t_1(s_2(q_2)) = s_1(t_1(q_2))$. In the left hand side, the second source cell of q_2 is, in this case, the transition $s_2(q_2) = q_1^1 = (q_0^1, q_0^2)$ and the first target cell of q_1^1 is q_0^2 (the only target cell because $s_2(q_2) \in Q_1$); this must be the same cell when taking the right hand side of the cubical law, i.e., the first target cell is $t_1(q_2) = q_1^2 = (q_0^2, q_0^3)$ and the first source of q_1^2 is q_0^2 .

We propose the language of *higher dimensional modal logic* for talking about concurrent systems. *HDML* follows the tradition and style of standard modal languages [14].

Definition 2.2 (higher dimensional modal logic). *A formula φ in higher dimensional modal logic is constructed using the grammar below, from a set Φ_B of atomic propositions, with $\phi \in \Phi_B$, which are combined using the Boolean symbols \perp and \rightarrow (from which all other standard propositional operations are generated), and using the modalities $\{\}$ and $\langle\rangle$.*

$$\varphi ::= \phi \mid \perp \mid \varphi \rightarrow \varphi \mid \{\}\varphi \mid \langle\rangle\varphi$$

We call $\{\}$ the *during modality* and $\langle\rangle$ the *after modality*. The intuitive reading of $\{\}\varphi$ is: “pick some event from the ones currently not running (must exist at least one not running) and start it; in the new configuration of the system (during which, one more event is concurrently executing) the formula φ must hold”. The intuitive reading of $\langle\rangle\varphi$ is: “pick some event from the ones currently running concurrently (must exist one running) and terminate it; in the new configuration of the system the formula φ must hold”. This intuition is formalized in the semantics of *HDML*.

The choice of our notation is biased by the intuitive usage of these modalities where the after modality talks about what happens after some event is terminated; in this respect being similar to the standard diamond modality of dynamic logic. Later, in Section 3.3, these modalities are decorated with labels. The during modality talks about what happens during the execution of some event and hence we adopt the notation of Pratt [15].

The models of *HDML* are higher dimensional structures together with a valuation function $\mathcal{V} : Q \rightarrow 2^{\Phi_B}$ which associates a set of atomic propositions to each cell (of any dimension). This means that \mathcal{V} assigns some propositions to each state of dimension 0, to each transition of dimension 1, to each square of dimension 2, to each cube of dimension 3, etc. Denote a model of *HDML* by $\mathcal{H} = (Q, \bar{s}, \bar{t}, l, \mathcal{V})$. A *HDML* formula is evaluated in a cell of such a model \mathcal{H} .

One may see the *HDML* models as divided into *levels*, each level increasing the concurrency complexity of the system; i.e., level Q_n increases the complexity compared to level Q_{n-1} by adding one more event (to have n events executing concurrently instead of $n - 1$). One can see Q_0 as having concurrency complexity 0 because there are no events executing there. The levels are linked together through the s_i and t_i maps. With this view in mind the during and after modalities should be understood as jumping from one level to the other; the $\{\}$ modality jumps one level up, whereas the $\langle\rangle$ modality jumps one level down.

Definition 2.3 (satisfiability). *Table 1 defines recursively the satisfaction relation \models of a formula φ wrt. a model \mathcal{H} in a particular n -cell q (for some arbitrary n); denote this as $\mathcal{H}, q \models \varphi$. The notions of satisfiability and validity are defined as usual.*

Both modalities have an existential flavor. In particular note that $\mathcal{H}, q_0 \not\models \langle\rangle\varphi$, for $q_0 \in Q_0$ a state, because there is no event executing in a state, and thus no event can be terminated. Similarly, for the during modality, $\mathcal{H}, q_n \not\models \{\}\varphi$ for any n -cell $q_n \in Q_n$ when all sets Q_k , with $n < k$, are empty (i.e., the family of sets Q is bounded by n). This says that there can be at most n events running at the same time, and when reaching this limit one cannot start another event and therefore $\{\}\varphi$ cannot be satisfied.

The universal correspondents of $\{\}$ and $\langle\rangle$ are defined in the usual style of modal logic. We denote these modalities by respectively $\llbracket\rrbracket\varphi$ and $\llbracket\rangle\varphi$; eg. $\llbracket\rrbracket\varphi \triangleq \neg\{\}\neg\varphi$. The intuitive reading of $\llbracket\rrbracket\varphi$ is: “pick any of the events currently running concurrently and after terminating it, φ must hold in the new configuration of the system”. Note that this modality holds trivially for any state $q_0 \in Q_0$, i.e., $\mathcal{H}, q_0 \models \llbracket\rrbracket\varphi$.

$\mathcal{H}, q \models \phi$	iff $\phi \in \mathcal{V}(q)$.
$\mathcal{H}, q \not\models \perp$	
$\mathcal{H}, q \models \phi_1 \rightarrow \phi_2$	iff when $\mathcal{H}, q \models \phi_1$ then $\mathcal{H}, q \models \phi_2$.
$\mathcal{H}, q \models \{\}\phi$	iff assuming $q \in Q_n$ for some n , $\exists q' \in Q_{n+1}$ s.t. $s_i(q') = q$ for some $1 \leq i \leq n+1$, and $\mathcal{H}, q' \models \phi$.
$\mathcal{H}, q \models \langle \rangle \phi$	iff assuming $q \in Q_n$ for some n , $\exists q' \in Q_{n-1}$ s.t. $t_i(q) = q'$ for some $1 \leq i \leq n$, and $\mathcal{H}, q' \models \phi$.

Table 1: Semantics for *HDML*.

In the rest of this section we prove that satisfiability for *HDML* is decidable using a variation of the filtration technique [14]. Then we give an axiomatic system for *HDML* and prove its soundness.

2.1 Decidability of *HDML*

The filtration for the states is the same as in the standard modal logic, but for cells of dimension 1 or higher we need to take care that the maps t and s in the filtration model remain maps and that they respect the cubical laws so that the filtration is still a *HDML* model. This can be done, but the filtration model is bigger than what is obtained in the case of standard modal logic. On top, the proof of the small model property (Theorem 2.13) is more involved due to the complexities of the definition of filtration given in Definition 2.6.

Definition 2.4 (subformula closure). *The subformula closure of a formula ϕ is the set of formulas $\mathcal{C}(\phi)$ defined recursively as:*

$$\begin{aligned}
\mathcal{C}(\phi) &\triangleq \{\phi\}, \text{ for } \phi \in \Phi_B \\
\mathcal{C}(\phi_1 \rightarrow \phi_2) &\triangleq \{\phi_1 \rightarrow \phi_2\} \cup \mathcal{C}(\phi_1) \cup \mathcal{C}(\phi_2) \\
\mathcal{C}(\{\}\phi) &\triangleq \{\{\}\phi\} \cup \mathcal{C}(\phi) \\
\mathcal{C}(\langle \rangle \phi) &\triangleq \{\langle \rangle \phi\} \cup \mathcal{C}(\phi)
\end{aligned}$$

The *size* of a formula (denoted $|\phi|$) is calculated by summing the number of Boolean and modal symbols with the number of atomic propositions and \perp symbols that appear in the formula. (All instances of a symbol are counted.)

Proposition 2.5 (size of the closure). *The size of the subformula closure of a formula ϕ is linear in the size of the formula; i.e., $|\mathcal{C}(\phi)| \leq |\phi|$.*

Proof. The proof is easy, using structural induction and observing that for the atomic formulas the size of the closure is exactly 1, the size of the formula. For a compound formula like $\{\}\phi$ the induction hypothesis says that $|\mathcal{C}(\phi)| \leq |\phi|$ which means $1 + |\mathcal{C}(\phi)| \leq 1 + |\phi|$. \square

Definition 2.6 (filtration). *Given a formula ϕ , we define below a relation \equiv (which is easily proven to be an equivalence relation) over the cells of a higher dimensional structure \mathcal{H} , where $q, q' \in Q_i$, for some $i \in \mathbb{N}$:*

$$q \equiv q' \text{ iff for any } \psi \in \mathcal{C}(\phi) \text{ then } (\mathcal{H}, q \models \psi \text{ iff } \mathcal{H}, q' \models \psi).$$

A filtration model \mathcal{H}^f of some structure \mathcal{H} through the closure set $\mathcal{C}(\phi)$ is the structure $(Q^f, s^f, t^f, l^f, \mathcal{V}^f)$:

$$\begin{aligned}
Q_n^f &\triangleq \{[q_n] \mid q_n \in Q_n\}, \text{ where } [q_n] \text{ is} \\
[q_0] &\triangleq \{q' \mid q_0 \equiv q'\} \text{ when } q_0 \in Q_0, \text{ otherwise,} \\
[q_n] &\triangleq \{q' \mid q_n \equiv q' \wedge t_i(q') \in [p_i] \wedge s_i(q') \in [p'_i] \\
&\quad \text{for all } 1 \leq i \leq n \text{ and for some fixed } [p_i], [p'_i] \in Q_{n-1}^f\}. \\
s_i^f([q_n]) &\triangleq [q_{n-1}] \text{ iff for all } p \in [q_n], s_i(p) \in [q_{n-1}]. \\
t_i^f([q_n]) &\triangleq [q_{n-1}] \text{ iff for all } p \in [q_n], t_i(p) \in [q_{n-1}]. \\
\mathcal{V}^f([q]) &\triangleq \mathcal{V}(q).
\end{aligned}$$

Lemma 2.7. Any two sets $[p], [q] \in Q_n^f$, for some $n \in \mathbb{N}$, are disjoint.

Proof. By induction on n .

The base case for $n = 0$ is easy as the definition of Q_0^f results in the equivalence classes on Q_0 generated by the equivalence relation \equiv , which are disjoint.

Inductive step: Consider $[p], [q] \in Q_n^f$, for which we assume that $\exists r \in Q_n$ with $r \in [p]$ and $r \in [q]$. From the definition we get (1) $q \equiv r \equiv p$ and, (2) for any $1 \leq i \leq n$ and some fixed $[p'_i], [q'_i] \in Q_{n-1}^f$, $t_i(r) \in [p'_i]$ and $t_i(r) \in [q'_i]$. By the induction hypothesis we know that $[p'_i]$ and $[q'_i]$ are disjoint, which, together with (2) before, implies that $[p'_i] = [q'_i]$ for all $1 \leq i \leq n$. Because of this and (1) it implies that $[q] = [p]$. Therefore we have proven that if two sets $[p], [q] \in Q_n^f$ have a cell in common then they must be the same. (Note that an analogous treatment of s_i is needed.) \square

Lemma 2.8.

1. The definitions of s_i^f and t_i^f are that of maps (as required in a higher dimensional structure).
2. The s_i^f and t_i^f respect the cubical laws of a higher dimensional structure.

Proof. For 1. we give the proof only for t_i^f , as the proof for s_i^f is analogous. We use *reductio ad absurdum* and assume, for some $[q] \in Q_n^f$, that $t_i^f([q]) = [p]$ and $t_i^f([q]) = [p']$ with $[p] \neq [p']$ and $[p], [p'] \in Q_{n-1}^f$. From the definition we have that for all $q \in [q]$ both $t_i(q) \in [p]$ and $t_i(q) \in [p']$. From Lemma 2.7 we know that $[p]$ and $[p']$ are disjoint and we know that t_i is a map (i.e., the outcome is unique), therefore we have the contradiction.

We have thus proven that for some input, t_i^f returns a unique output. It now remains to show that t_i^f is a total map; i.e., that for any input $[q] \in Q_n^f$, with $n > 0$, it returns some output $t_i^f([q]) = [p]$. Since $[q]$ is not empty then it has at least one $q \in [q]$ and cf. Definition 2.6, $t_i(q) \in [q']$ for some fixed $[q'] \in Q_{n-1}^f$. By Definition 2.6, if there are other $q_n \in [q]$ then $t_i(q_n)$ is also part of the fixed $[q']$. Thus, $\forall q_n \in [q] : t_i(q_n) \in [q']$ meaning that $[q']$ is the outcome we are looking for $t_i^f([q])$. The same reasoning goes analogous for s_i^f .

For 2. we have to prove, for some arbitrary chosen $[q] \in Q_n^f$ and for any $1 \leq i < j \leq n$ that

$$t_i^f(t_j^f([q])) = t_{j-1}^f(t_i^f([q])).$$

(Note that t_i^f on the left side is different than the t_i^f on the right side, as the left one is applied to elements of Q_{n-1}^f whereas the right one is applied to elements of Q_n^f .) The other three kinds of cubical laws are treated analogous only that one needs to reason with the s_i maps too.

Assume, wlog. because the opposite assumption would follow analogous reasoning, that $t_i^f(t_j^f([q])) = [q_{n-2}]$ with $[q_{n-2}] \in Q_{n-2}^f$. This leads to considering that $t_j^f([q]) = [q_{n-1}]$ with $[q_{n-1}] \in Q_{n-1}^f$, and $t_i^f([q_{n-1}]) = [q_{n-2}]$. From the definition we have both:

- (1) $\forall q \in [q] : t_j(q) \in [q_{n-1}]$,
- (2) $\forall q \in [q_{n-1}] : t_i(q) \in [q_{n-2}]$.

Therefore, from the two we have that

- (3) $\forall q \in [q] : t_i(t_j(q)) \in [q_{n-2}]$.

We want to prove that $[q_{n-2}] = t_{j-1}^f(t_i^f([q]))$, for which we can assume that $t_i^f([q]) = [q'_{n-1}]$ for some $[q'_{n-1}] \in Q_{n-1}^f$. Therefore, it amounts to proving that $t_{j-1}^f([q'_{n-1}]) = [q_{n-2}]$. For this it is enough to find some $p \in [q'_{n-1}]$ s.t. $t_{j-1}(p) \in [q_{n-2}]$, because by the Definition 2.6 (of the t_i maps) it means that $\forall p \in [q'_{n-1}]$ it holds that $t_{j-1}(p) \in [q_{n-2}]$, i.e., our desired result.

From the assumption we have that $\forall q \in [q] : t_i(q) \in [q'_{n-1}]$. Pick one of these $t_i(q)$ and claim this to be the $p \in [q'_{n-1}]$ we are looking for. From the cubical laws for the initial \mathcal{H} model we know that for any $q \in [q]$, $t_i(t_j(q)) = t_{j-1}(t_i(q)) = t_{j-1}(p)$. Because of (3) we have that $t_{j-1}(p) \in [q_{n-2}]$, and thus our claim is proven; i.e., t_{j-1} applied to the element $t_i(q)$ that we picked from $[q'_{n-1}]$, is in $[q_{n-2}]$. \square

Corollary 2.9 (filtration is a model). *The filtration \mathcal{H}^f of a model \mathcal{H} through a closure set $\mathcal{C}(\varphi)$ is a higher dimensional structure (i.e., is still a HDML model).*

Proof. Essentially, the proof amounts to showing that the definitions of s_i^f and t_i^f are that of *maps* and that they respect the *cubical laws* which were done in Lemma 2.8. \square

Lemma 2.10 (sizes of filtration sets). *Each set Q_n^f of the filtration \mathcal{H}^f obtained in Definition 2.6 has finite size which depends on the size of the formula φ used in the filtration; more precisely each Q_n^f is bounded from above by $2^{|\varphi| \cdot N}$ where $N = n! \cdot \sum_{k=0}^n \frac{2^k}{(n-k)!}$.*

Proof. The case for 0 is simple as the number of equivalence classes of Q_0 can be maximum the number of subsets of the subformula closure $\mathcal{C}(\varphi)$ which is $2^{|\varphi|}$.

The case for $n = 1$ is based on the size of Q_0^f . Each of the $2^{|\varphi|}$ equivalence classes in which Q_1^f can be divided may have infinitely many cells. Any such equivalence class can still be broken into smaller subsets depending on the maps t_1 and s_1 . Because t_1 can have outcome in any of the $[q_0] \in Q_0^f$, we get a first split into $2^{|\varphi|}$ subdivisions. For each of these we can still split it into $2^{|\varphi|}$ more subdivisions because of s_1 . We thus get a maximum of $2^{|\varphi|} \cdot (2^{|\varphi|})^{2 \cdot 1}$ for Q_1^f . For the general case of n we need to consider all maps t_i, s_i , that means $2 \cdot n$ maps. For each of these maps we split the $2^{|\varphi|}$ possible initial equivalence classes according to the size of Q_{n-1}^f . Thus we get a maximum of $2^{|\varphi|} \cdot (|Q_{n-1}^f|)^{2 \cdot n}$ subdivisions. Calculating this series gives the bound on the size of Q_n^f as being $2^{|\varphi| \cdot N}$ where $N = n! \cdot \sum_{k=0}^n \frac{2^k}{(n-k)!}$. \square

As a side remark, the size of O_n^f is more than double exponential in the dimension n , but is less than triple exponential. More precisely, for N , the sum is bounded from above by $(n+1) \cdot 2^n$ which makes N the order of $n! \cdot (n+1) \cdot 2^n$. We know that $n!$ grows faster than exponential, but not too fast; more precisely, using Stirling's approximation of $n!$ we have that $\lg(n!) = \Theta(n \cdot \lg(n))$ making $n! \cdot (n+1) \cdot 2^n = (n+1) \cdot 2^{n+\lg(n!)}$ of order $(n+1) \cdot 2^{\Theta(n \cdot (\lg(n)+1))}$. Therefore, $|O_n^f|$ is bounded by $2^{|\varphi| \cdot (n+1) \cdot 2^{\Theta(n \cdot (\lg(n)+1))}}$ (where we consider $|\varphi|$ to be a constant, and hence, not contributing to the bound).²

Lemma 2.11 (filtration lemma). *Let \mathcal{H}^f be the filtration of \mathcal{H} through the closure set $\mathcal{C}(\varphi)$, as in Definition 2.6. For any formula $\psi \in \mathcal{C}(\varphi)$ and any cell $q \in \mathcal{H}$, we have $\mathcal{H}, q \models \psi$ iff $\mathcal{H}^f, [q] \models \psi$.*

Proof. By induction on the structure of the formula ψ .

Base case: For $\psi = \phi \in \Phi_B$ is immediate from the definition of \mathcal{V}^f .

Inductive step: The case for \rightarrow is straightforward making use of the induction hypothesis because the set $\mathcal{C}(\varphi)$ is closed under subformulas.

Take now $\psi = \langle \rangle \psi'$ and we prove that $\mathcal{H}, q \models \langle \rangle \psi'$ iff $\mathcal{H}^f, [q] \models \langle \rangle \psi'$. Considering the *only if* implication we assume that (cf. definition of satisfiability from Table 1) $\exists q' \in Q_{n-1} : t_i(q) = q' \wedge q' \models \psi'$ for some $1 \leq i \leq n$, and have to prove that $\exists [p] \in Q_{n-1}^f : t_i^f([q]) = [p] \wedge [p] \models \psi'$. Because $q \in [q]$ and $t_i(q) = q'$, using the definition of $[q]$ it implies that for all $q \in [q]$ is that $t_i(q) \in [q']$ which, by the definition of t_i^f , implies that $t_i^f([q]) = [q']$. (Thus we have found the $[p] = [q'] \in Q_{n-1}^f$.) From the induction hypothesis we have that $\mathcal{H}, q' \models \psi'$ implies that $\mathcal{H}^f, [q'] \models \psi'$. This ends the proof.

Consider now the *if* implication and assume $\exists [p] \in Q_{n-1}^f : t_i^f([q]) = [p] \wedge [p] \models \psi'$ for some $1 \leq i \leq n$. From the definition of t_i^f we have that $t_i(q) \in [p]$; which is the same as picking some $p' \in [p]$ with $t_i(q) = p'$. From the induction hypothesis we know that $\mathcal{H}^f, [p] \models \psi'$ iff $\mathcal{H}, p \models \psi'$ for any $p \in [p]$ (in particular $\mathcal{H}, p' \models \psi'$). Thus $\exists p' \in Q_{n-1} : t_i(q) = p' \wedge \mathcal{H}, p' \models \psi'$ for some $1 \leq i \leq n$, finishing the proof.

When we take $\psi = \{ \} \psi'$ we use analogous arguments as in the proof of $\langle \rangle \psi'$. In this case we work with the definition of s_i^f and we look for cells of higher dimension (instead of lower dimension). \square

²This discussion is for $n > 0$ because \lg is undefined for 0.

We define two *degrees of concurrency* of a formula φ : the *upwards concurrency* (denoted $|\varphi|_{uc}$) and *downwards concurrency* (denoted $|\varphi|_{dc}$). The degree of upwards concurrency counts the maximum number of nestings of the during modality $\{\}$ that are not compensated by a $\langle \rangle$ modality. (E.g., the formula $\{\}\{\}\phi \vee \{\}\phi'$ has the degree of upwards concurrency equal to 2, the same as $\{\}\langle \rangle\{\}\{\}\phi$.) The formal definition of $|\cdot|_{uc}$ is:

$$\begin{aligned} |\perp|_{uc} &\triangleq |\phi|_{uc} \triangleq 0, \text{ for } \phi \in \Phi_B \\ |\varphi_1 \rightarrow \varphi_2|_{uc} &\triangleq \max(|\varphi_1|_{uc}, |\varphi_2|_{uc}) \\ |\{\}\varphi|_{uc} &\triangleq 1 + |\varphi|_{uc} \\ |\langle \rangle\varphi|_{uc} &\triangleq \max(0, |\varphi|_{uc} - 1) \end{aligned}$$

The definition of the degree of downwards concurrency $|\cdot|_{dc}$ is symmetric to the one above in the two modalities; i.e., interchange the modalities in the last two lines. Note that $|\varphi|_{uc} + |\varphi|_{dc} \leq |\varphi|$. The next result offers a safe reduction of a model where we remove all cells which have dimension greater than some constant depending on the formula of interest.

Lemma 2.12 (concurrency boundedness). *If a HDML formula φ is satisfiable, $\mathcal{H}, q \models \varphi$ with $q \in Q_k$, then it exists a model with all the sets Q_m , with $m > |\varphi|_{uc} + k$, empty, which satisfies the formula.*

Proof. By induction on the structure of the formula φ .

Base case: For $\phi \in \Phi_B$ and \perp the evaluation is in the same cell q and thus all the cells of dimension higher than k are not important and can be empty.

Inductive step: For $\varphi_1 \rightarrow \varphi_2$ the semantics says that whenever $\mathcal{H}, q \models \varphi_1$ then $\mathcal{H}, q \models \varphi_2$. From the induction hypothesis we have that all cells of dimension greater than $k + |\varphi_1|_{uc}$ (respectively $k + |\varphi_2|_{uc}$) are not important for checking φ_1 (respectively φ_2). Thus it is a safe approximation to consider all the cells of at most dimension $\max(k + |\varphi_1|_{uc}, k + |\varphi_2|_{uc}) = k + |\varphi_1 \rightarrow \varphi_2|_{uc}$ and all sets Q_m of greater dimension can be empty.

For $\{\}\varphi$ the semantics says that we need to check the formula φ in cells of dimension one greater, i.e., $q_{k+1} \models \varphi$. From the induction hypothesis we know that for checking $q_{k+1} \models \varphi$ it is enough to have only cells of most dimension $k + 1 + |\varphi|_{uc} = k + |\{\}\varphi|_{uc}$ (where all other cells can be removed).

For $\langle \rangle\varphi$ the semantics says that we need to check $q_{k-1} \models \varphi$, that is, in cells of immediately lower dimension. For this, the induction hypothesis says that we need to consider cells of dimension at most $k - 1 + |\varphi|_{uc}$ which is the same as $k + (|\varphi|_{uc} - 1)$. When $|\varphi|_{uc} = 0$ then k is a safe approximation and from the definition of the $|\cdot|_{uc}$ it is the same as $k + |\langle \rangle\varphi|_{uc}$. Otherwise, when $|\varphi|_{uc} > 0$, the definition of $|\cdot|_{uc}$ tells us that $k + (|\varphi|_{uc} - 1)$ is exactly $k + |\langle \rangle\varphi|_{uc}$. \square

Notation: The formula $\langle \rangle\phi \wedge \langle \rangle\neg\phi$ expresses that there can be terminated at least two different events (in other words, the cell in which the formula is evaluated to true has dimension at least two). Similarly the formula $\langle \rangle(\phi \wedge \neg\phi') \wedge \langle \rangle(\neg\phi \wedge \neg\phi') \wedge \langle \rangle(\neg\phi \wedge \phi')$ says that there are at least three events that can be terminated. For each $i \in \mathbb{N}^*$ one can write such a formula to say that there are at least i events that can be terminated. Denote such a formula by $\langle \rangle^i$. Also define $\langle \rangle^i\varphi$ as i applications of the $\langle \rangle$ modality to φ (i.e., $\langle \rangle \dots \langle \rangle\varphi$ where $\langle \rangle$ appears i times). Similar, for the during modality denote $\{\}^i$ the formula that can start i different events, and by $\{\}^i\varphi$ the i applications of $\{\}$ to φ .

Theorem 2.13 (small model property). *If a HDML formula φ is satisfiable then it is satisfiable on a finite model with no more than $\sum_{n=0}^{|\varphi|} 2^{|\varphi| \cdot N}$ cells where $N = n! \cdot \sum_{k=0}^n \frac{2^k}{(n-k)!}$.*

Proof. Assume that there exists a model \mathcal{H} and a cell $q_l \in Q_l$ in this model for which $\mathcal{H}, q_l \models \varphi$. We can prove that there exists a (maybe different) model \mathcal{H}' and a cell q'_l that satisfy φ but which $l < |\varphi| - |\varphi|_{uc}$. We do this by induction on the structure of φ .

Base case: when $\varphi = \phi \in \Phi_B$. The semantics needs to look only at the valuations, and by the assumption, the valuation of q_l in \mathcal{H} satisfies φ . Hence we can just use one cell model where we attach this satisfying valuation to it. Therefore level Q_0 is enough; hence $l = 0 < |\phi| - |\phi|_{uc} = 1 - 0$.

Inductive step: when $\varphi = \varphi_1 \rightarrow \varphi_2$. By the semantics it means that whenever φ_1 is satisfied in q_l also φ_2 is. But by the induction hypothesis it means that $l < |\varphi_1| - |\varphi_1|_{uc}$ and also $l < |\varphi_2| - |\varphi_2|_{uc}$. Therefore it is a safe approximation to take l to be the maximum of the two: $l < \max(|\varphi_1| - |\varphi_1|_{uc}, |\varphi_2| - |\varphi_2|_{uc})$. We have to show that $l < |\varphi| - |\varphi|_{uc}$ and we do this by showing that $\max(|\varphi_1| - |\varphi_1|_{uc}, |\varphi_2| - |\varphi_2|_{uc}) < |\varphi| - |\varphi|_{uc}$. By expanding the definition on the right we get the inequality $\max(|\varphi_1| - |\varphi_1|_{uc}, |\varphi_2| - |\varphi_2|_{uc}) < |\varphi_1| + |\varphi_2| + 1 - \max(|\varphi_1|_{uc}, |\varphi_2|_{uc})$. This amounts to showing that $\max(|\varphi_1| - |\varphi_1|_{uc}, |\varphi_2| - |\varphi_2|_{uc}) + \max(|\varphi_1|_{uc}, |\varphi_2|_{uc}) < |\varphi_1| + |\varphi_2| + 1$. Denote the quantity $|\varphi_1| - |\varphi_1|_{uc} = A$ and $|\varphi_2| - |\varphi_2|_{uc} = B$ and hence have $|\varphi_1| = A + |\varphi_1|_{uc}$ and $|\varphi_2| = B + |\varphi_2|_{uc}$. Thus the inequality translates to $\max(A, B) + \max(|\varphi_1|_{uc}, |\varphi_2|_{uc}) < A + |\varphi_1|_{uc} + B + |\varphi_2|_{uc} + 1$. Since both A and B (also the other quantities in the inequality) are positive the result is obvious as $\max(A, B) < A + B$ (as being one of the summands) and $\max(|\varphi_1|_{uc}, |\varphi_2|_{uc}) < |\varphi_1|_{uc} + |\varphi_2|_{uc}$.

When $\varphi = \{\}\varphi_1$ the semantics says that exists $q_{l+1} \in Q_{l+1}$ where φ_1 holds. The inductive hypothesis says that $l + 1 < |\varphi_1| - |\varphi_1|_{uc}$. This means that $l < |\varphi_1| - |\varphi_1|_{uc} - 1 = |\varphi_1| - |\{\}\varphi_1|_{uc} < |\varphi_1| + 1 - |\{\}\varphi_1|_{uc} = |\{\}\varphi_1| - |\{\}\varphi_1|_{uc}$.

When $\varphi = \langle \rangle \varphi_1$ the semantics says that exists $q_{l-1} \in Q_{l-1}$ where φ_1 holds. From the inductive hypothesis we have $l - 1 < |\varphi_1| - |\varphi_1|_{uc}$. This means that $l < |\varphi_1| + 1 - |\varphi_1|_{uc} = |\langle \rangle \varphi_1| - |\varphi_1|_{uc}$. Because $\max(0, |\varphi_1|_{uc} - 1) < |\varphi_1|_{uc}$ it means that $|\langle \rangle \varphi_1| - |\varphi_1|_{uc} < |\langle \rangle \varphi_1| - \max(0, |\varphi_1|_{uc} - 1)$ hence $l < |\langle \rangle \varphi_1| - |\langle \rangle \varphi_1|_{uc}$.

From the above we can safely assume $l = |\varphi| - |\varphi|_{uc}$.

From Lemma 2.12 we know that we need to consider only the sets Q_n for $n \leq l + |\varphi|_{uc} = |\varphi|$, and all other sets of Q are empty. From Lemma 2.11 we know that we can build a filtration model \mathcal{H}^f s.t. the formula φ is still satisfiable and, by Lemma 2.10, we know that all the sets Q_n^f have a finite number of cells. Thus we are safe if we sum up all the cells in all the Q_n^f , with $n \leq |\varphi|$. \square

Corollary 2.14 (decidability). *Deciding the satisfiability of a HDML formula φ is done in space at most $\sum_{n=0}^{|\varphi|} 2^{|\varphi| \cdot N}$ where N is defined in Theorem 2.13.*

2.2 Axiomatic system for HDML

In the following we give an axiomatic system for *HDML* and prove it to be sound. This system corrects the one in [1]. In Table 2 we give a set of axioms and rules of inference for *HDML*. If a formula is *derivable* in this axiomatic system we write $\vdash \varphi$. We say that a formula φ is derivable from a set of formulas S iff $\vdash \psi_1 \wedge \dots \wedge \psi_n \rightarrow \varphi$ for some $\psi_1, \dots, \psi_n \in S$ (we write equivalently $S \vdash \varphi$). A set of formulas S is said to be *consistent* if $S \not\vdash \perp$, otherwise it is said to be *inconsistent*. A consistent set S is called *maximal* iff all sets S' , with $S \subset S'$, are inconsistent.

Proposition 2.15 (theorems). *The following are derivable in the axiomatic system of Table 2:*

Axiom schemes:

- (A1) All instances of propositional tautologies.
- (A2) $\{\} \perp \leftrightarrow \perp$ (A2') $\langle \rangle \perp \leftrightarrow \perp$
- (A3) $\{(\varphi \vee \varphi')\} \leftrightarrow \{\varphi\} \vee \{\varphi'\}$ (A3') $\langle (\varphi \vee \varphi') \rangle \leftrightarrow \langle \varphi \rangle \vee \langle \varphi' \rangle$
- (A4) $\llbracket \varphi \rrbracket \leftrightarrow \neg \{\} \neg \varphi$ (A4') $\llbracket \varphi \rrbracket \leftrightarrow \neg \langle \rangle \neg \varphi$
- (A5) $\langle i \rangle \rightarrow \langle i \rangle^{\top} \quad \forall i \in \mathbb{N}^*$
- (A6) $\langle \rangle^2 \top \rightarrow (\langle \rangle \llbracket \varphi \rrbracket \rightarrow \llbracket \langle \rangle \varphi \rrbracket)$
- (A7) $\{\} \llbracket \varphi \rrbracket \rightarrow \llbracket \{\} \varphi \rrbracket$ (A7') $\langle \rangle \llbracket \varphi \rrbracket \rightarrow \llbracket \langle \rangle \varphi \rrbracket$
- (A8) $\{\} \langle i \rangle^{\top} \rightarrow \llbracket \{\} \langle i \rangle^{\top} \rrbracket \quad \forall i \in \mathbb{N}$ (A8') $\langle \rangle \langle i \rangle^{\top} \rightarrow \llbracket \langle \rangle \langle i \rangle^{\top} \rrbracket \quad \forall i \in \mathbb{N}$
- (A9) $\langle i \rangle^{\top} \rightarrow \llbracket \langle i \rangle^{\top} \rrbracket \quad \forall i \in \mathbb{N}$ (A9') $\{\} \langle i \rangle^{\top} \rightarrow \langle i \rangle^{\top} \quad \forall i \in \mathbb{N}$
- (A10) $\{\} \{\} \langle \rangle \varphi \rightarrow \{\} \langle \rangle \{\} \varphi$ (A10') $\{\} \langle \rangle \langle \rangle \varphi \rightarrow \langle \rangle \{\} \langle \rangle \varphi$

Inference rules:

- (R1) $\frac{\varphi \quad \varphi \rightarrow \varphi'}{\varphi'} \quad (\text{MP})$
- (R2) $\frac{\varphi \rightarrow \varphi'}{\{\} \varphi \rightarrow \{\} \varphi'} \quad (\text{D})$ (R2') $\frac{\varphi \rightarrow \varphi'}{\langle \rangle \varphi \rightarrow \langle \rangle \varphi'} \quad (\text{D}')$
- (R3) Uniform variable substitution.

Table 2: Axiomatic system for *HDML*.

- $$\begin{aligned} &\vdash \{(\varphi \rightarrow \varphi')\} \rightarrow (\{\varphi\} \rightarrow \{\varphi'\}) & (1) \\ &\vdash \langle (\varphi \rightarrow \varphi') \rangle \rightarrow (\langle \varphi \rangle \rightarrow \langle \varphi' \rangle) & (2) \\ &\vdash \langle \rangle^2 \top \rightarrow (\langle \rangle \llbracket \varphi \wedge \langle \rangle \neg \varphi \rrbracket \rightarrow \perp) & (3) \\ &\vdash (\langle \rangle \langle \rangle \varphi \wedge \langle \rangle \neg \varphi) \rightarrow \langle \rangle^3 \top & (4) \\ &\vdash \llbracket \perp \rrbracket \rightarrow (\langle \rangle \varphi \rightarrow \llbracket \varphi \rrbracket) & (5) \\ &\vdash \langle \rangle \top \rightarrow (\{\} \llbracket \varphi \rrbracket \rightarrow \langle \rangle \{\} \varphi) & (6) \\ &\vdash \{\} \top \rightarrow (\langle \rangle \llbracket \varphi \rrbracket \rightarrow \{\} \langle \rangle \varphi) & (7) \\ &\vdash \llbracket \langle \rangle \top \rrbracket & (8) \\ &\vdash \langle \rangle \llbracket \perp \rrbracket \rightarrow \llbracket \perp \rrbracket & (9) \\ &\vdash \{\} \top \rightarrow \llbracket \{\} \top \rrbracket & (10) \\ &\vdash \{\} \top \wedge \langle \rangle \top \rightarrow \langle \rangle \{\} \top & (11) \\ &\vdash \langle \rangle \top \rightarrow (\{\} \langle \rangle \top \rightarrow \langle \rangle \{\} \top) & (12) \\ &\vdash \{(\langle \rangle \phi \wedge \langle \rangle \neg \phi)\} \rightarrow (\langle \rangle \{\} \phi \vee \langle \rangle \{\} \neg \phi) & (13) \\ &\vdash \{\} \{\} \langle \rangle \phi \rightarrow \{\} \langle \rangle \{\} \phi & (14) \\ &\vdash \{\} \{\} \{\} \langle \rangle \phi \rightarrow \{\} \langle \rangle \{\} \{\} \phi & (15) \\ &\vdash \{\} \{\} \langle \rangle \{\} \phi \rightarrow \{\} \langle \rangle \{\} \{\} \phi & (16) \\ &\vdash \llbracket \llbracket \varphi \rrbracket \rrbracket \rightarrow \llbracket \llbracket \varphi \rrbracket \rrbracket & (17) \\ &\vdash \llbracket \llbracket \varphi \rrbracket \rrbracket \rightarrow \llbracket \llbracket \varphi \rrbracket \rrbracket & (18) \end{aligned}$$

Moreover, one can use the following derived rules:

$$\frac{\frac{\varphi}{\llbracket \varphi \rrbracket}, \quad \frac{\varphi}{\llbracket \varphi \rrbracket},}{\frac{\varphi \rightarrow \varphi'}{\llbracket \varphi \rrbracket \rightarrow \llbracket \varphi' \rrbracket}, \quad \frac{\varphi \rightarrow \varphi'}{\llbracket \varphi \rrbracket \rightarrow \llbracket \varphi' \rrbracket}.$$

Proof. The first two theorems are derivable as in standard modal logic only using the standard axioms (A2)-(A3'). The derived rules are also as in standard modal logic. The theorem (3) is a consequence of (A6): $\langle \rangle^2 \top \rightarrow \langle \rangle \llbracket \varphi \wedge \langle \rangle \neg \varphi \rightarrow \langle \rangle^2 \top \rightarrow \llbracket \varphi \wedge \langle \rangle \neg \varphi \xrightarrow{SML} \langle \rangle (\langle \rangle \varphi \wedge \langle \rangle \neg \varphi) \xrightarrow{SML} \langle \rangle \langle \rangle (\varphi \wedge \neg \varphi) \xrightarrow{(A2')} \perp$. The theorem (5) uses the contrapositive of axiom (A5): $\llbracket \varphi \rrbracket \perp \leftrightarrow \neg \langle \rangle \langle \rangle \top \rightarrow \neg (\langle \rangle 2) \leftrightarrow \neg (\langle \rangle \varphi \wedge \langle \rangle \neg \varphi) \leftrightarrow (\langle \rangle \varphi \rightarrow \llbracket \varphi \rrbracket)$. The theorem (4) uses axiom (A6). The theorem (6) is a consequence of (A7): from propositional reasoning we have $\langle \rangle \top \rightarrow (\{ \} \llbracket \varphi \rrbracket \rightarrow \langle \rangle \{ \} \varphi) \leftrightarrow (\{ \} \llbracket \varphi \wedge \langle \rangle \top \rightarrow \langle \rangle \{ \} \varphi)$, and using (A7) we have $\{ \} \llbracket \varphi \wedge \langle \rangle \top \xrightarrow{(A7)} \llbracket \{ \} \varphi \wedge \langle \rangle \top \rightarrow \langle \rangle \{ \} \varphi$. The theorem (7) is derivable in an analogous way as the one above only that we use axiom (A7'). The theorem (8) is just the instantiation of axiom (A9) when $i = 0$ (i.e., $\langle \rangle^0 \top \triangleq \top$). The theorem (9) is a consequence of (A7'): $\langle \rangle \llbracket \varphi \rrbracket \perp \xrightarrow{(A7')} \llbracket \langle \rangle \varphi \rrbracket \perp \xrightarrow{(A2')} \llbracket \varphi \rrbracket \perp$. The theorem (10) is a consequence of the theorem (9) by contraposition. The theorem (11) is derivable from theorem (8). The theorem (12) is derivable from theorem (11). The theorem (13) is derivable from theorem (11) after using axiom (A5) and axiom (A9') instantiate to $i = 1$: $\{ \} \langle \rangle 2 \xrightarrow{(A5)} \{ \} \langle \rangle^2 \top \equiv \{ \} \top \wedge \{ \} \langle \rangle^2 \top \xrightarrow{(A9')} \{ \} \top \wedge \langle \rangle \top \xrightarrow{(11)} \langle \rangle \{ \} \top \xrightarrow{prop} \langle \rangle \{ \} (\varphi \vee \neg \varphi) \xrightarrow{SML} \langle \rangle (\{ \} (\varphi) \vee \{ \} (\neg \varphi)) \xrightarrow{SML} \langle \rangle \{ \} (\varphi) \vee \langle \rangle \{ \} (\neg \varphi)$. Theorem (14) follows either from axiom (A10) by the D' rule or from axiom (A10') by the D rule. Theorem (16) is an instantiation of axiom (A10). Theorem (15) needs twice the application of axiom (A10) and the D rule. We need here the application of the axiom two times because we move the $\langle \rangle$ modality two times over $\{ \}$, whereas for the other theorems we move the modality only once. The theorems (17) and (18) are just the contrapositives of axioms (A10) respectively (A10'). \square

Exercise 2.1. A challenge is to prove the validity of:

$$\langle \rangle (p \wedge \llbracket \neg p \rrbracket) \wedge \langle \rangle (\neg p \wedge \llbracket \neg p \rrbracket) \wedge \langle \rangle \langle \rangle p \rightarrow \langle \rangle^4 \top$$

This challenge is related to theorem 2.15.(4). A general version of this challenge should be possible, where one can deduce $\langle \rangle^i \top$ from $\langle \rangle \langle \rangle p$ and $i - 1$ distinct formulas $\langle \rangle (\phi_i \wedge \llbracket \neg p \rrbracket)$ which contradict on the ϕ_i components.

Before proving soundness we should have some intuition about the non-standard axioms (A5) to (A10'). First consider the axioms (A6) to (A7') which relate to the cubical laws.

- Axiom (A6) embodies the cubical law $t_i(t_j(q)) = t_{j-1}(t_i(q))$ (i.e., the cubical law where α is instantiated to t and β to t). This axiom is to be checked only for cell of dimension 2 or higher (i.e., $\langle \rangle^2 \top$ holds).
- The two axioms (A7) and (A7') relate to the cubical laws where α and β are instantiated differently, one to s and the other to t ; e.g., $s_i(t_j(q)) = t_{j-1}(s_i(q))$. We included both axioms (A7) and (A7') for symmetry reasons, but it is clear that one can be obtained from the other by contraposition.

The other axioms talk about the dimensions of the cells and about the division of the cells into layers \mathcal{Q}_n .

- Axiom (A5) $\langle \rangle i \rightarrow \langle \rangle^i \top$ says that if in a cell there can be terminated at least i different events then this means that this cell has dimension at least i (i.e., one can go i levels down by $\langle \rangle^i \top$). This is natural because the dimension of a cell is given by the number of events that are currently executing concurrently.
- Axiom (A9) $\langle \rangle^i \top \rightarrow \llbracket \varphi \rrbracket \langle \rangle^i \top$ has two purposes. In the basic variant (for $i = 0$ it becomes $\llbracket \varphi \rrbracket \langle \rangle \top$) it says that in any cell, however one starts an event then one can also terminate an event. In the general form the axiom says that from some level i when going one level up (by starting an event) and then one level down (by terminating an event) we always end up on the same level i ; i.e., we end in a cell of the same dimension like

the cell that it started in. Axiom (A9') intuitively finds out the level of the current cell. If one can start and then can terminate an event in a cell of at least dimension i then the current cell also has dimension at least i .

- Axiom (A8) intuitively says that if from a cell we can start an event and reach a cell of some concurrency complexity (given by the $\langle \rangle^i \top$) then any way of starting an event from this cell ends up in cells of the same complexity. Though similar in nature, axiom (A8') can be seen intuitively as saying that if one t map of the current cell ends up in a cell of dimension at least i then all the t maps end up in the same dimension. These two axioms relate with the part of the definition of the *HDA* where all the s_i and t_i maps for some n are defined on the same domain and codomain.
- Axioms (A10) and (A10') are somehow related to the notion of homotopy (see eg. [3, ch.7.4]) or to the ways one can walk (i.e., the *paths* on a *HDA*, to be defined later) on the *HDAs* using the *HDML* modalities (or in other terms, these axioms are related to the histories of an event). One may reach a cell from another cell in a *HDA* in different ways and the notion of homotopy says that all these ways are considered equivalent. Take the example of the square (cell of dimension 2) from Figure 1 where the state in the upper-right corner can be reached from the cell in the lower-left corner in more than one way.

In this setting axioms (A10) and (A10') basically say that instead of going through the inside of a square one can go on one of its sides. In other words, instead of going through a cell of higher dimension one can go only through cells of lower dimensions. Particular to our example from Figure 1 the axiom (A10) says that when going from the lower-left corner through the inside of the square one can instead go through one of the lower or left sides and reach the same place. The other axiom (A10') says that for reaching the upper-right corner, instead of going through its inside one can just take one of its upper or right sides.

Note also the theorems (14)-(16) which involve four *HDML* modalities stacked one on top of the other. These are theorems of the two axioms (A10) and (A10') which involve only three modalities. In particular note the converse implication of (14) which is not a theorem. This says intuitively that one cannot infer from just being able to walk on the edges of a square that the square is filled in, i.e., that true concurrency is present. This makes *HDML* powerful enough for the distinction between true concurrency and interleaving.

Remark that a natural counterpart (using the $\{\}$ modality in place of $\langle \rangle$) of the axiom (A6) is $\{\}\{\}\{\}\{\}\varphi \rightarrow \{\}\{\}\{\}\varphi$ (which appeared in the short paper version [1]). But this “axiom” is broken by the fact that *HDAs* allow choices. This formula would be valid only when working inside a single full cube (i.e., no choices, just concurrency), as would be the case when representing Mazurkiewicz traces as *HDAs*.

Theorem 2.16 (soundness). *The axiomatic system of Table 2 is sound; i.e., $\forall \varphi : \vdash \varphi \Rightarrow \models \varphi$.*

Proof. For soundness of the axiomatic system it is enough to prove that the axioms (A5) to (A10') are valid.

We start with axiom (A6) and assume $\mathcal{H}, q_n \models \langle \rangle [\varphi]$ for some $q_n \in Q_n$ and $n \geq 2$ because of the assumption $\langle \rangle^2 \top$. This means that exists some $q_{n-1} \in Q_{n-1}$ s.t. $t_k(q_n) = q_{n-1}$ for some $1 \leq k \leq n$ with $\mathcal{H}, q_{n-1} \models [\varphi]$, and from this it means that for any $1 \leq l \leq n-1$, $\mathcal{H}, t_l(q_{n-1}) \models \varphi$. We need to show that $\mathcal{H}, q_n \models [\langle \rangle \varphi]$. This means that for any $m \neq k$ we have to find a $1 \leq m' \leq n-1$ s.t. $\mathcal{H}, t_{m'}(t_m(q_n)) \models \varphi$.³ This is easy by applying the cubical law, considering wlog. $m < k$, $t_m(t_k(q_n)) = t_{k-1}(t_m(q_n))$.⁴ Thus, the $m' = k-1$ for which trivially $1 \leq k-1 \leq n-1$. From the assumption we showed that we have $\mathcal{H}, t_m(t_k(q_n)) \models \varphi$ and hence $\mathcal{H}, t_{k-1}(t_m(q_n)) \models \varphi$.

For axiom (A7) assume $\mathcal{H}, q_n \models \{\} [\varphi]$ with $q_n \in Q_n$. This means that exists $q_{n+1} \in Q_{n+1}$ and $1 \leq k \leq n+1$ s.t. $s_k(q_{n+1}) = q_n$ and $\mathcal{H}, q_{n+1} \models [\varphi]$. Further, this implies that for any $1 \leq i \leq n+1$, $\mathcal{H}, t_i(q_{n+1}) \models \varphi$. We want to prove that $\mathcal{H}, q_n \models [\{\} \varphi]$, which amounts to showing that for some arbitrary $1 \leq m \leq n$ with $t_m(q_n) = q_{n-1}$ we can find an $1 \leq l \leq n$ and $q'_n \in Q_n$ s.t. $s_l(q'_n) = q_{n-1}$ and $\mathcal{H}, q'_n \models \varphi$. We assume that it exists at least one t_m to work with, for otherwise the formula $[\{\} \varphi]$ holds trivially. We achieve the goal using the cubical laws: if $m < k$ then

³We do not consider the k because the case for $m = k$ is trivial from the assumption above, where we know that for t_k and any t_l it is the case that $\mathcal{H}, t_l(t_k(q_n)) \models \varphi$; and because we are at least on the layer 2 it means that there exists at least one t_l .

⁴We can apply the cubical laws because we are working with cells of dimension at least 2. For the other case of $m > k$ we get $m' = k$ by using a corresponding cubical law.

consider the cubical law $t_m(s_k(q_{n+1})) = s_{k-1}(t_m(q_{n+1}))$ and set $l = k - 1$ and $q'_n = t_m(q_{n+1})$ for which we know from above that $\mathcal{H}, t_m(q_{n+1}) \models \varphi$; otherwise if $k \leq m$ (which also means that $k \leq n$) then consider the cubical law $s_k(t_{m+1}(q_{n+1})) = t_m(s_k(q_{n+1}))$ and set $l = k$ and $q'_n = t_{m+1}(q_{n+1})$ (where $m + 1 \leq n + 1$) for which we know that $\mathcal{H}, t_{m+1}(q_{n+1}) \models \varphi$.

For (A7') we can just use propositional reasoning and argue its validity by contraposition with axiom (A7) above. Nevertheless, we want to also give here a model theoretic argument similar to the above. Thus, assume $\mathcal{H}, q_n \models \langle \rangle \llbracket \rrbracket \varphi$ with $q_n \in Q_n$. This means that exists $1 \leq k \leq n$ and q_{n-1} s.t. $t_k(q_n) = q_{n-1}$ and $\mathcal{H}, q_{n-1} \models \llbracket \rrbracket \varphi$, which means that for any q'_n with $s_i(q'_n) = q_{n-1}$ for some $1 \leq i \leq n$ we have $\mathcal{H}, q'_n \models \varphi$. We want to prove that $\mathcal{H}, q_n \models \llbracket \rrbracket \langle \rangle \varphi$ which amounts to showing that for some arbitrary q_{n+1} , with $s_m(q_{n+1}) = q_n$ for some $1 \leq m \leq n + 1$, we can find an $1 \leq l \leq n + 1$ and a q''_n s.t. $t_l(q_{n+1}) = q''_n$ and $\mathcal{H}, q''_n \models \varphi$. We use the cubical laws: if $k < m$ then consider the cubical law $t_k(s_m(q_{n+1})) = s_{m-1}(t_k(q_{n+1}))$ and set $l = k$ and $q''_n = t_k(q_{n+1})$ for which we have said before that $\mathcal{H}, t_l(q_{n+1}) \models \varphi$ because there is the s_{m-1} that reaches a cell which satisfies $\llbracket \rrbracket \varphi$; otherwise if $m \leq k$ then consider the cubical law $s_m(t_{k+1}(q_{n+1})) = t_k(s_m(q_{n+1}))$ and set $l = k + 1$ and $q''_n = t_{k+1}(q_{n+1})$ for which it holds that $\mathcal{H}, t_l(q_{n+1}) \models \varphi$ because $\mathcal{H}, s_m(t_l(q_{n+1})) \models \llbracket \rrbracket \varphi$.

For axiom (A9) assume $\mathcal{H}, q_n \models \langle \rangle^i \top$ which means that $n \geq i$. Even more, $\langle \rangle^i \top$ holds in any cell $q_n \in Q_n$ of dimension n . We need to prove that $\mathcal{H}, q_n \models \llbracket \rrbracket \langle \rangle^i \top$. The proof is trivial when there is no q_{n+1} with $s_j(q_{n+1}) = q_n$. Therefore, we need to prove that for any q_{n+1} with $s_j(q_{n+1}) = q_n$, for some $1 \leq j \leq n + 1$, $\mathcal{H}, q_{n+1} \models \langle \rangle \langle \rangle^i \top$. Because $q_{n+1} \in Q_{n+1}$ then it must have at least one t map that links it with some cell $q'_n \in Q_n$ on the lower level. In q'_n the formula $\langle \rangle^i \top$ holds and thus we finished the proof.

For axiom (A9') assume $\mathcal{H}, q_n \models \{ \} \langle \rangle \langle \rangle^i \top$ which means that exists $q_{n+1} \in Q_{n+1}$ with $s_j(q_{n+1}) = q_n$ for some $1 \leq j \leq n + 1$ s.t. $\mathcal{H}, q_{n+1} \models \langle \rangle \langle \rangle^i \top$. This means that $n + 1 \geq i + 1$ and thus $n \geq i$. Therefore, for any $q'_n \in Q_n$ the formula $\langle \rangle^i \top$ holds because we can go at least i levels down and find any cell satisfying \top , hence $\langle \rangle^i \top$ holds also in $q_n \in Q_n$.

Axiom (A8) can actually be derived from axioms (A9) and (A9') as follows: for $i > 1$ then $\{ \} \langle \rangle^i \top \xrightarrow{(A9')} \langle \rangle^{i-1} \top \xrightarrow{(A9)} \llbracket \rrbracket \langle \rangle^i \top$; whereas for $i = 1$ it is just an instantiation of axiom (A9) for $i = 0$. As we did for axiom (A7') we leave these so that the reader has a more intuitive understanding of the apparent symmetries of these formulas.

Nevertheless, we give also a model-theoretic argument, hence assume $\mathcal{H}, q_n \models \{ \} \langle \rangle^i \top$. This means that exists q_{n+1} and $1 \leq j \leq n + 1$ s.t. $s_j(q_{n+1}) = q_n$ and $\mathcal{H}, q_{n+1} \models \langle \rangle^i \top$. This means that the dimension of q_{n+1} is greater than i , i.e., $n + 1 \geq i$. We want to prove that $\mathcal{H}, q_n \models \llbracket \rrbracket \langle \rangle^i \top$ which amounts to showing that for any $q'_{n+1} \in Q_{n+1}$ with $s_j(q'_{n+1}) = q_n$ for some $1 \leq j \leq n + 1$ we have $\mathcal{H}, q'_{n+1} \models \langle \rangle^i \top$. But we know from before that the dimension of q'_{n+1} is at least i ; this means that we can go down at least i levels and on the lowest level any cell models \top . Hence we have $\mathcal{H}, q'_{n+1} \models \langle \rangle^i \top$.

For axiom (A8') we use a similar argument as in the proof based on the semantics of $\langle \rangle$ and $\llbracket \rrbracket$ this time.

For (A5) consider that $\mathcal{H}, q \models \langle \rangle^i$ which means that there exist i different cells q^j with $1 \leq j \leq i$ which are the result of the application of a t map to q . Because t is a map it means that there exist at least i different maps t_j with $1 \leq j \leq i$ that are applied to q . Therefore, q is of dimension at least i which means that we can go i levels down (by using an inductive argument). This makes the formula $\langle \rangle^i \top$ true at q .

For (A10) assume $\mathcal{H}, q_n \models \{ \} \{ \} \langle \rangle \varphi$ which by the definition of the semantics it means that $\exists q_{n+1} \in Q_{n+1}, k \leq n + 1 : s_k(q_{n+1}) = q_n$ and $\exists q_{n+2} \in Q_{n+2}, i \leq n + 2 : s_i(q_{n+2}) = q_{n+1}$ and $\exists q'_{n+1} \in Q_{n+1}, j \leq n + 1 : t_j(q_{n+2}) = q'_{n+1}$ and $\mathcal{H}, t_j(q_{n+2}) \models \varphi$. We want to prove that $\mathcal{H}, q_n \models \{ \} \langle \rangle \{ \} \varphi$. This amounts to finding three cells $q_{n+1}^a \in Q_{n+1}$, $q_n^b \in Q_n$, and $q_{n+1}^c \in Q_{n+1}$ s.t. $s_l(q_{n+1}^a) = q_n$, $t_m(q_{n+1}^a) = q_n^b$, and $s_n(q_{n+1}^c) = q_n^b$ and $\mathcal{H}, q_{n+1}^c \models \varphi$. We treat three cases depending on i and j .

Case when $j < i$ then choose $m = j$, $n = i - 1$, $k = l$, and $q_{n+1}^a = q_{n+1}$ hence finding the cubical law $t_m(s_i(q_{n+2})) = s_n(t_j(q_{n+2}))$ which makes $t_j(q_{n+2}) = q_{n+1}^c$ and hence, the desired $\mathcal{H}, q_{n+1}^c \models \varphi$ follows from the initial $\mathcal{H}, t_j(q_{n+2}) \models \varphi$.

Case when $j > i$ then choose $m = j - 1$, $n = i$, $k = l$, and $q_{n+1}^a = q_{n+1}$ hence finding the cubical law $s_n(t_j(q_{n+2})) = t_m(s_i(q_{n+2}))$ which makes $t_j(q_{n+2}) = q_{n+1}^c$ and hence, the desired $\mathcal{H}, q_{n+1}^c \models \varphi$ follows as before.

Case when $i = j$ then it is not enough to work only with the i and j as the cubical laws do not apply any more. But there are ways depending on k . We need two cases. When $k < j$ consider $l = j - 1$, $m = j - 1$, $n = k$, and $q_{n+1}^a = s_k(q_{n+2})$ as coming from the cubical law $s_k(s_j(q_{n+2})) = s_l(s_k(q_{n+2}))$. Using a second cubical law $s_k(t_j(q_{n+2})) = t_m(s_k(q_{n+2})) = q_n^b$ we obtain $q_{n+1}^c = q_{n+1}'$ and hence the desired $\mathcal{H}, q_{n+1}^c \models \phi$. Otherwise, when $k \geq j$ then choose $l = j$, $m = j$, $n = k$ and $q_{n+1}^a = s_{k+1}(q_{n+2})$ as coming from the cubical law $s_l(s_{k+1}(q_{n+2})) = s_k(s_j(q_{n+2}))$. Using as second cubical law $t_m(s_{k+1}(q_{n+2})) = s_k(t_j(q_{n+2})) = q_n^b$ we obtain $q_{n+1}^c = q_{n+1}'$ and hence the desired result as before.

For (A10') assume $\mathcal{H}, q_n \models \{\}\langle\rangle\phi$ which by the definition of the semantics it means that $\exists q_{n+1} \in Q_{n+1}, i \leq n+1 : s_i(q_{n+1}) = q_n$ and $\exists q_n' \in Q_n, j \leq n : t_j(q_{n+1}) = q_n'$ and $\exists q_{n-1} \in Q_{n-1}, k \leq n-1 : t_k(q_n') = q_{n-1}$ and $\mathcal{H}, q_{n-1} \models \phi$. We want to prove that $\mathcal{H}, q_n \models \langle\rangle\{\}\langle\rangle\phi$. This amounts to finding three cells $q_{n-1}^a \in Q_{n-1}$, $q_n^b \in Q_n$, and $q_{n-1}^c \in Q_{n-1}$ s.t. $t_m(q_n) = q_{n-1}^a$, $s_n(q_n^b) = q_{n-1}^a$, and $t_l(q_n^b) = q_{n-1}^c$, and $\mathcal{H}, q_{n-1}^c \models \phi$. We again treat three cases depending on i and j .

Case when $i < j$ then choose $m = j - 1$, $n = i$, $l = k$, and $q_n^b = q_n'$ and get q_{n-1}^a from the cubical law $s_n(t_j(q_{n+1})) = t_m(s_i(q_{n+1})) = q_{n-1}^a$. Since $q_{n-1}^c = t_l(q_n^b) = t_k(q_n') = q_{n-1}$ we get our desired result $\mathcal{H}, q_{n-1}^c \models \phi$.

Case when $i > j$ then choose $m = j$, $n = i - 1$, $l = k$, and $q_n^b = q_n'$ and get q_{n-1}^a from the cubical law $t_m(s_i(q_{n+1})) = s_n(t_j(q_{n+1})) = q_{n-1}^a$. We get our desired result $\mathcal{H}, q_{n-1}^c \models \phi$ as before.

Case when $i = j$ requires two subcases after k as the cubical laws are not applicable to i and j anymore. We follow a similar reasoning as we did for (A10). When $k < j$ then choose $l = j - 1$ and have $q_n^b = t_k(q_{n+1})$ and $q_{n-1}^c = q_{n-1}$ from the cubical law $q_{n-1} = t_k(t_j(q_{n+1})) = t_l(t_k(q_{n+1}))$. To connect everything consider the cubical law $t_m(s_i(q_{n+1})) = s_n(t_k(q_{n+1}))$ giving $m = k$ and $n = j - 1$. When $k \geq j$ then choose $l = j$ and have $q_n^b = t_{k+1}(q_{n+1})$ and $q_{n-1}^c = q_{n-1}$ from the cubical law $t_l(t_{k+1}(q_{n+1})) = t_k(t_j(q_{n+1})) = q_{n-1}$. And all is connected right through the cubical law $s_n(t_{k+1}(q_{n+1})) = t_m(s_j(q_{n+1}))$ giving $m = k$ and $n = j$. \square

Theorem 2.17 (compactness failure). *The HDML with the semantics of Table 1 does not have the compactness property.*

Proof. Compactness says that for any infinite set of formulas Γ if all the finite subsets $S \subset \Gamma$ are satisfiable than the original Γ is satisfiable.

The compactness failure for *HDML* is witnessed by the following infinite set of formulas:

$$\Gamma = \{\langle\rangle^i \top \mid i \in \omega\}.$$

Any finite subset $S = \{\langle\rangle^i \top \mid i \leq n\}$ of Γ is satisfiable on a model \mathcal{H}_n which has $Q_n \neq \emptyset$ in any cell $q_n \in Q_n$ of dimension n ; i.e., $\mathcal{H}_n, q_n \models \langle\rangle^i \top$ for all $\langle\rangle^i \top \in S$.

On the other hand the infinite Γ is not satisfiable on any pointed model, i.e., at a single point. For assume there exists a model \mathcal{H} and some cell $q \in Q_m$ for some level m where all formulas $\phi \in \Gamma$ are satisfiable $\mathcal{H}, q \models \phi$. But this is not possible as the formula $\langle\rangle^{m+1} \top$ does not hold on any cell from level Q_m or any level below. This is because when stripping off one $\langle\rangle$ we go one level down cf. the semantics; and we cannot go down more than m levels, cf. $q \in Q_m$ but we need to strip $m+1$ times the after operator $\langle\rangle$. No matter on which level we choose the point cell q in a model there will always be a formula in Γ that will not hold, because of the infiniteness of Γ (also regardless of the infiniteness of the model that we choose).

Intuitively, the compactness failure is due to the fact that the models of *HDML* are bounded below in their levels and *HDML* has a modality that goes down the levels (i.e., the after modality $\langle\rangle$). \square

3 Examples of Encodings into Higher Dimensional Modal Logic

This section serves to exemplify ways of using *HDML*. One may encode other logics for different concurrency models as restrictions of *HDML*; in this respect we study the relation of *HDML* with standard modal logic, with CTL, ISTL (a branching time temporal logic over configuration structures), and with linear time temporal logic over

Mazurkiewicz traces LTrL. Another way of using *HDML* is as a general logical framework for studying properties of concurrency models and their interrelation. This is done by finding the appropriate restrictions of *HDA* and *HDML* and investigating their relations and axiomatic presentations.

3.1 Encoding standard modal logic into HDML

Lemma 3.1 (Kripke structures). *The class of Kripke structures is captured by the class of higher dimensional structures where all sets Q_n , for $n > 1$, are empty.*

Proof. Essentially this result is found in [3]. A *HDA* $K = (Q_0, Q_1, s_1, t_1, l)$ is a special case of *HDAs* where all $Q_n = \emptyset$ for $n > 1$. This is the class of *HDAs* that encode Kripke frames. Because Q_2 (and all other cells of higher dimension) is empty there are no cubical laws applicable. Therefore, there is no geometric structure on K . Moreover, the restriction on the labeling function l is not applicable (as Q_2 is empty). Add to such a *HDA* a valuation function \mathcal{V} to obtain a Kripke model $(Q_0, Q_1, s_1, t_1, l, \mathcal{V})$. \square

Proposition 3.2 (axiomatization of Kripke *HDAs*). *The class of higher dimensional structures corresponding to Kripke structures (from Lemma 3.1) is axiomatized by:*

$$\models \{\}\{\}\{\}\perp \quad (19)$$

Proof. For any *HDA* \mathcal{H} and any $q \in Q$ a cell of any dimension, we prove the double implication: $\mathcal{H} \models \{\}\{\}\{\}\perp$ iff \mathcal{H} is as in Lemma 3.1.

For the *if* direction if $q \in Q_1$ then the axiom holds trivially because there are no cells on Q_2 , hence $\mathcal{H}, q \models \{\}\{\}\{\}\perp$ holds and also $\{\}\{\}\perp$. When $q \in Q_0$ the axiom holds because for any $q' \in Q_1$ with $s_1(q') = q$ it is the case that $\mathcal{H}, q' \models \{\}\{\}\perp$ because there are no $q'' \in Q_2$ cf. Lemma 3.1.

For the *only if* direction consider a \mathcal{H} for which the axiom holds (i.e., for any cell $q \in Q$ then $\mathcal{H}, q \models \{\}\{\}\{\}\perp$); we need to show that any Q_n with $n > 1$ is empty. Assume the opposite, that there exists $q_n \in Q_n$ with $n > 1$. This means that there is a sequence $s_1(\dots s_i(q_n)) = q_0$ of source maps that ends in a cell $q_0 \in Q_0$ of dimension 0. But $\mathcal{H}, q_0 \models \{\}\{\}\{\}\perp$, which means that there cannot be this sequence of source maps unless q_n is of dimension at most 1. This is a contradiction and hence the proof is finished. \square

Theorem 3.3 (standard modal logic). *Consider the syntactic definition*

$$\Diamond \varphi \triangleq \{\}\langle \rangle \varphi.$$

The language of standard modal logic uses only \Diamond and is interpreted only over higher dimensional structures as defined in Lemma 3.1 and only in cells of Q_0 .

Proof. First we check that we capture exactly the semantics of standard modal logic; $\mathcal{H}, q_0 \models \Diamond \varphi$ iff $\mathcal{H}, q_0 \models \{\}\langle \rangle \varphi$ iff $\exists q' \in Q_1$ s.t. $s_1(q') = q_0$ and $\mathcal{H}, q' \models \langle \rangle \varphi$ iff $\exists q'_0 \in Q_0$ s.t. $t_1(q') = q'_0$ and $\mathcal{H}, q'_0 \models \varphi$. This is the same as $\exists q'_0 \in Q_0$ reached in “one transition” from q_0 and $\mathcal{H}, q'_0 \models \varphi$. (We go only through one transition cell $q' \in Q_1$.)

Clearly, with the axiom of Proposition 3.2, $\mathcal{H}, q_n \not\models \Diamond \varphi$ for any $q_n \in Q_n$ for any $n \geq 1$. Therefore, $\Diamond \varphi$ makes sense only interpreted in states from Q_0 .

Second we check that the axioms of standard modal logic for \Diamond hold in our axiomatic system. Clearly $\Diamond \perp \leftrightarrow \perp$; just apply (A2') and then (A2) to $\{\}\langle \rangle \perp$. It is easy to see that $\Box \varphi \leftrightarrow \neg \Diamond \neg \varphi$ as $\neg \{\}\langle \rangle \neg \varphi \xleftrightarrow{(A4)} \{\}\neg \langle \rangle \neg \varphi \xleftrightarrow{(A4')} \{\}\{\}\varphi$ and the semantic of $\Box \varphi$ is the right one, i.e., for any $q'_0 \in Q_0$, reached through some transition $q' \in Q_1$, is the case that $\mathcal{H}, q'_0 \models \varphi$. We prove now that $\Diamond(\varphi \vee \varphi') \leftrightarrow \Diamond \varphi \vee \Diamond \varphi'$. This is because $\{\}\langle \rangle(\varphi \vee \varphi') \xleftrightarrow{(A3')} \{\}\langle \rangle \varphi \vee \{\}\langle \rangle \varphi' \xleftrightarrow{(A3)} \{\}\langle \rangle \varphi \vee \{\}\langle \rangle \varphi' \xleftrightarrow{def} \Diamond \varphi \vee \Diamond \varphi'$.

It is easy to see how we recover the corresponding inference rule for \Diamond . We thus have all the axiomatic system of standard modal logic and the proof is finished. \square

Remark that the axioms (A5)-(A10') particular to *HDML* are trivially satisfied for all states or transitions (i.e., cells of dimension 0 or 1). This means that for these cells these axioms do not impose any constraints. One can easily check that for each of the axioms (A5)-(A10'), which are implications, either the first formula does not hold or the second formula holds trivially. In fact, in the axiomatic system of Table 2 with the new axiom (19) added, one cannot prove formulas where the same existential modality is stacked twice or more (like $\{\}\{\}\dots$ or $\langle\rangle\langle\rangle\dots$). In fact, any such formula is provably unsatisfiable. This is also a reason for using the syntactic definition for the diamond from Theorem 3.3.

3.2 Adding an Until operator and encoding standard temporal logic

The *basic* temporal logic is the logic with only the *eventually* operator (and the dual *always*). This language is expressible in the standard modal logic [14]. It is known that the *Until* operator adds expressiveness (*eventually* and *always* operators can be encoded with *Until* but not the other way around).

The *Until* operator cannot be encoded in *HDML* because of the local behavior of the *during* and *after* modalities; similar arguments as in modal logic about expressing *Until* apply to *HDML* too. The *Until* modality talks about the whole model (about all the configurations of the system) in an existential manner. More precisely, the *Until* says that there must exist some configuration in the model, reachable from the configuration where *Until* is evaluated, satisfying some property ϕ , and in all the configurations on all/some of the paths reaching the ϕ configuration some other property ψ must hold. Hence we need a notion of *path* in a *HDA*.

Definition 3.4 (paths in HDAs). A simple step in a HDA is either $q_{n-1} \xrightarrow{s_i} q_n$ with $s_i(q_n) = q_{n-1}$ or $q_n \xrightarrow{t_i} q_{n-1}$ with $t_i(q_n) = q_{n-1}$, where $q_n \in Q_n$ and $q_{n-1} \in Q_{n-1}$ and $1 \leq i \leq n$. A path $\pi \triangleq q^0 \xrightarrow{\alpha^0} q^1 \xrightarrow{\alpha^1} q^2 \xrightarrow{\alpha^2} \dots$ is a sequence of single steps $q^j \xrightarrow{\alpha^j} q^{j+1}$, with $\alpha^j \in \{s_i, t_i\}$. We say that $q \in \pi$ iff $q = q^j$ appears in one of the steps in π . The first cell in a path is denoted $st(\pi)$ and the ending cell in a finite path is $en(\pi)$. We call a cell q' reachable from some other cell q , and denote by $q \rightarrow^* q'$, iff $\exists \pi : st(\pi) = q \wedge en(\pi) = q'$. Overload the notation $\pi \rightarrow^* \pi'$ to mean that the path π' extends π , with the usual definition.

There are two main kinds of *Until* operator that can be defined on a branching structure like *HDA*: one is in the style of linear time temporal logic [16]; and the other in the style of computation tree logic (CTL). These two kinds are found defined also over Mazurkiewicz traces or configuration structures. There are proofs that the CTL style of defining the *Until* yields undecidability both on traces [17] and on configuration structures [18, 10] and all these three proofs use different techniques, i.e., encoding a different undecidable problem. On the other hand the LTL style of definition of *Until* over traces is decidable as part of LTrL [9]; see also the related decidable definition part of the TrPTL logic [7].

In the same spirit as done for temporal logic we boost the expressiveness of *HDML* by defining an *Until* operator over higher dimensional structures. We define both styles of *Until* operators. We then show how the standard LTL logic (with its until operator interpreted over Kripke structures) is encoded into the *HDML* framework. For the CTL-like definition we discuss if and how the details of the undecidability proofs over Mazurkiewicz traces can be done in the setting of *HDML*. Note that the proofs in [17, 10] lack many of the details. We concentrate on the proof using the Post correspondence problem from [10].

Definition 3.5 (CTL-like Until operator). Define an Until operator $\phi \mathcal{U}^c \phi'$, in the style of CTL, which is interpreted over a HDA in a cell as below:

$$\begin{aligned} \mathcal{H}, q \models \phi \mathcal{U}^c \phi' &\text{ iff } \exists \pi \in \mathcal{H} \text{ s.t. } st(\pi) = q \wedge en(\pi) = q', \\ &\mathcal{H}, q' \models \phi', \text{ and } \forall q'' \in \pi, q'' \neq q' \text{ then } \mathcal{H}, q'' \models \phi. \end{aligned}$$

Definition 3.6 (LTL-like Until operator). Define an Until operator $\phi \mathcal{U}^l \phi'$, in the style of LTL, which is interpreted over a HDA in a cell as below:

$$\begin{aligned} \mathcal{H}, q \models \phi \mathcal{U}^l \phi' &\text{ iff } \exists q' \in \mathcal{H} \text{ s.t. } q \rightarrow^* q' \wedge \mathcal{H}, q' \models \phi', \\ &\text{and } \forall \pi \in \mathcal{H}, \forall q'' \in \pi : st(\pi) = q \wedge en(\pi) = q' \wedge q'' \neq q' \\ &\text{then } \mathcal{H}, q'' \models \phi. \end{aligned}$$

The Definition 3.6 of \mathcal{U}^l is in the style of LTL in the sense that it looks only at one (concurrent) execution of the system ignoring choices (in the sense of *HDA*). The Definition 3.5 of \mathcal{U}^c is more refined because it looks at a single linearization of a concurrent execution; and it is branching in the sense that it is not confined to one single concurrent execution, but the linearization may cross boundaries of concurrent runs, i.e., taking choices.

Proposition 3.7 (modeling CTL Until). *The CTL Until modality is encoded syntactically by $\varphi \exists \mathcal{U} \varphi' \triangleq (\varphi \vee \langle \rangle \top) \mathcal{U}^c (\varphi' \wedge \neg \langle \rangle \top)$ when $\exists \mathcal{U}$ is interpreted only in states of Kripke HDAs as in Lemma 3.1.*

Proof. Essential for the proof is the fact that $\exists \mathcal{U}$ is interpreted over restricted HDAs which model Kripke structures. Precisely, they have only cells of dimension 0 (the states) and 1 (the transitions), and moreover, we know which are states because the formula $\neg \langle \rangle \top$ holds in all and only the cells of dimension 0. Therefore, the right formula of the $\exists \mathcal{U}$ is evaluated only in states because $(\varphi' \wedge \neg \langle \rangle \top)$ can never hold in a cell of dimension greater than 0. Moreover, the transitions are not important for valuating the φ because the formula $\langle \rangle \top$ is always true in a transition (because any transition has a target state). On the other hand the formula $\langle \rangle \top$ is never true in a state and hence the φ has to be true so that the whole left part of the until to hold.

For this proof we only concentrate on showing that the semantics of the $\exists \mathcal{U}$ corresponds to the well known CTL semantics. Thus, we want to show that $\mathcal{H}, q_0 \models \varphi \exists \mathcal{U} \varphi'$ is the same as saying that exists a finite sequence of states $q_0^1, \dots, q_0^k \in Q_0$ with $q_0^1 = q_0$, $\mathcal{H}, q_0^k \models \varphi'$, $\mathcal{H}, q_i^1 \models \varphi$ for all $1 \leq i < k$, and for any $1 < i \leq k$ q_0^i is reachable through a single transition from q_0^{i-1} . By the definition in the statement, $\mathcal{H}, q_0 \models \varphi \exists \mathcal{U} \varphi'$ is the same as $\mathcal{H}, q_0 \models (\varphi \vee \langle \rangle \top) \mathcal{U}^c (\varphi' \wedge \neg \langle \rangle \top)$. By the semantics of \mathcal{U}^c from Definition 3.5 we know that $\exists \pi$ a path in \mathcal{H} , which goes only through cells of dimension 0 or 1 because \mathcal{H} models a Kripke structure cf. Lemma 3.1, hence π is of the form q_0, q_1, q'_0, \dots ; and moreover, we also have that $st(\pi) = q_0 \wedge en(\pi) = q'$, $\mathcal{H}, q' \models (\varphi' \wedge \neg \langle \rangle \top)$, and $\forall q'' \in \pi, q'' \neq q'$ then $\mathcal{H}, q'' \models (\varphi \vee \langle \rangle \top)$. Clearly $q' \in Q_0$ because $\neg \langle \rangle \top$ must hold in q' and hence φ' holds in a state, i.e., $\mathcal{H}, q' \models \varphi'$. It remains to show that in all q'' which are states (i.e., those $q_0^k \in Q_0$) we have that $\mathcal{H}, q'' \models \varphi$. But we know that $\mathcal{H}, q'' \not\models \langle \rangle \top$ because q'' , being a cell of dimension 0, has no t map. Therefore, using $\mathcal{H}, q'' \models (\varphi \vee \langle \rangle \top)$ from before, we have that $\mathcal{H}, q'' \models \varphi$.

Note that for the full CTL a universal correspondent of \mathcal{U}^c must be defined over HDAs, but we do not go into these details here. \square

3.3 Partial order models and their logics in HDML

This section is mainly concerned with Mazurkiewicz traces [19] as a model of concurrency based on partial orders, because of the wealth of logics that have been developed for it [7, 9]. Higher dimensional automata are more expressive than most of the partial orders models (like Mazurkiewicz traces, pomsets [20], or event structures [21]) as studied in [22, 3]. In particular, an extensive part of [3] is devoted to showing how Petri nets are representable as some class of higher dimensional automata. The works of [22, 6, 3] show (similar in nature) how event structures can be encoded in higher dimensional automata. Mazurkiewicz traces are a particular class of event structures, precisely defined in [23]. We use this presentation, as a restricted partial order, of Mazurkiewicz traces.

In the following we give definitions and standard results on partial orders, event structures, and Mazurkiewicz traces which are needed for the development of the higher dimensional modal logic for these models, in particular for Mazurkiewicz traces. In few words, we isolate the class of higher dimensional automata corresponding to Mazurkiewicz traces (and to partial orders or event structures in general) as the models of the *HDML*. Then we restrict *HDML* to get exactly the logics over Mazurkiewicz traces (we focus on the logics presented in [9, 24]) and over the more general partial orders called communicating sequential agents in [25] (like ISTL of [18, 10]).

Definition 3.8 (partial orders). *A partially ordered set (or poset) is a set E equipped with a partial order \leq , (E, \leq) . The history of an element $e \in E$ (denoted $\downarrow e$) is $\downarrow e = \{e' \mid e' \leq e\}$. The notion of history is extended naturally to a set of elements $C \subseteq E$ (denoted $\downarrow C$). A configuration is a finite and history closed set of elements (i.e., $C = \downarrow C$). Denote by \mathcal{C} the set of all configurations. (Obviously, \emptyset , and $\downarrow e$, for any $e \in E$, are configurations.) The immediate successor relation $\leq \subseteq E \times E$ is defined as $e \leq e'$ iff $e \neq e'$ and $e \leq e'$ and $\forall e'' \in E$, $e \leq e'' \leq e'$ implies $e = e''$ or*

$e' = e''$. A Σ -labeled poset (E, \leq, λ) is a poset with a labeling function $\lambda : E \rightarrow \Sigma$ which maps each element to a label from Σ . Define a transition relation on the configurations of a labeled poset as $\longrightarrow \subseteq \mathcal{C} \times \Sigma \times \mathcal{C}$ given by $C \xrightarrow{a} C'$ iff $\exists e \in E$ s.t. $\lambda(e) = a$ and $e \notin C$ and $C' = C \cup \{e\}$.

When one sees the elements of E as the *events* of a system, the labels can be seen as the names of the actions that the events are instances of.

Definition 3.9 (Mazurkiewicz traces). Consider a symmetric and irreflexive independence relation $I \subseteq \Sigma \times \Sigma$ and its complement $D = \Sigma \times \Sigma \setminus I$, called the dependence relation. Mazurkiewicz traces are labeled posets restricted by the independence relation as follows:

$$\begin{aligned} \forall e \in E, \quad \downarrow e \text{ is finite,} \\ \forall e, e' \in E, \quad e < e' \Rightarrow (\lambda(e), \lambda(e')) \in D, \\ \forall e, e' \in E, \quad (\lambda(e), \lambda(e')) \in D \Rightarrow e \leq e' \text{ or } e' \leq e. \end{aligned}$$

Definition 3.10 (event structures). Consider a symmetric and irreflexive relation $\# \subseteq E \times E$. This conflict relation is added to a poset to form an event structure $(E, \leq, \#)$ where the following restrictions apply:

$$\begin{aligned} \forall e, e', e'' \in E, \quad e \# e' \text{ and } e' \leq e'' \text{ implies } e \# e'', \\ \forall e, e' \in E, \quad e \in C \text{ and } e \# e' \text{ implies } e' \notin C. \end{aligned}$$

An event structure is called finitary iff $\forall e \in E, \downarrow e$ is finite.

The second constraint on event structures says that the configurations of an event structure are conflict-free. Define the relation of concurrency for an event structure to be:

$$co = E \times E \setminus (\# \cup \leq \cup \leq^{-1}).$$

Proposition 3.11 (families of configurations). A finitary event structure $(E, \leq, \#)$ is uniquely determined by its family of configurations \mathcal{C}_E (denoted (E, \mathcal{C}_E)).

Proof. This result is found in [6]. We summarize here the results leading to it.

The two relations $e \leq e'$ and $e \# e'$ are mutually exclusive, because, otherwise, the set $\downarrow e'$ would not be a configuration (because of the second constraint of Definition 3.10).

If two events e, e' do not appear together in any configuration of \mathcal{C}_E then $e \# e'$ ($e \# e'$ iff $\nexists C \in \mathcal{C}_E$ s.t. $e, e' \in C$).

If in any configuration where e' exists, e exists too then $e \leq e'$ ($e \leq e'$ iff $\forall C \in \mathcal{C}_E, e' \in C \Rightarrow e \in C$). \square

We usually use a labeled poset and work with labeled event structures $(E, \leq, \#, \lambda)$, or $(E, \mathcal{C}_E, \lambda)$ when using their corresponding family of configurations.

Proposition 3.12 (traces as event structures). Any Mazurkiewicz trace, as in Definition 3.9, corresponds to a trace configuration structure, which is a labeled event structure $(E, \mathcal{C}_E, \lambda)$ with an empty conflict relation that respects the following restriction:

$$\lambda \text{ is a nice labeling and context-independent,}$$

where nice labeling means

$$\forall e, e' \in E, \quad \lambda(e) = \lambda(e') \Rightarrow e \leq e' \text{ or } e' \leq e$$

and context-independent means

$$\forall a, b \in \Sigma, \quad (\lambda^{-1}(a) \times \lambda^{-1}(b)) \cap co \neq \emptyset \Rightarrow (\lambda^{-1}(a) \times \lambda^{-1}(b)) \cap \leq = \emptyset.$$

Proof. This result is essentially found in [7, 23]. We remind how one gets the independence relation of a Mazurkiewicz trace from a trace configuration structure:

$$I = \{(a, b) \mid (\lambda^{-1}(a) \times \lambda^{-1}(b)) \cap co \neq \emptyset\}.$$

\square

One can view a configuration as a valuation of events $E \rightarrow \{0, 1\}$, and thus we can view an event structure as a valuation $f_E : 2^E \rightarrow \{0, 1\}$, which selects only those configurations that make the event structure.

The terminology that we adopt now steams from the Chu spaces representation of HDAs [22, 6]. We fix a set E , which for our purposes denotes events. Consider the class of HDAs which have a single hypercube of dimension

$|E|$, hence each event represents one dimension in the *HDA*. This hypercube is denoted 3^E , in relation to 2^E , because in the *HDA* case each event may be in three phases, *not started*, *executing*, and *terminated* (as opposed to only terminated or not started). The valuation from before becomes now $E \rightarrow \{0, \frac{1}{2}, 1\}$, where $\frac{1}{2}$ means executing. The set of three values is linearly ordered $0 < \frac{1}{2} < 1$ to obtain an *acyclic HDA* [6], and all cells of 3^E are ordered by the natural lifting of this order pointwise. The dimension of a cell is equal to the number of $\frac{1}{2}$ in its corresponding valuation.

Notation: In the context of a single hypercube 3^E we denote the cells of the cube by lists of $|E|$ elements $e_1 e_2 \dots e_{|E|}$ where each e_i takes values in $\{0, \frac{1}{2}, 1\}$ and represents the status of the i^{th} event of the *HDA*.

With the above conventions, the cells of dimension 0 (i.e., the states of the *HDA*) are denoted by the corresponding valuation restricted to only the two values $\{0, 1\}$; and correspond to the configurations of an event structure. The set of states of such a *HDA* is partially ordered by the order $<$ we defined before. In this way, from the hypercube 3^E we can obtain any family of configurations \mathcal{C}_E by removing all 0-dimensional cells that represent a configuration $C \notin \mathcal{C}_E$.⁵ By Proposition 3.11 we can reconstruct the event structure.

In Definition 2.3 the interpretation of the during and after modalities of *HDML* did not take into consideration the labeling of the *HDA*. The labeling was used only for defining the geometry of concurrency of the *HDA*. Now we make use of this labeling function in the semantics of the labeled modalities of Definition 3.14. But first we show how the labeling extends to cells of any dimension.

Definition 3.13 (general labeling). *Because of the condition $l(s_i(q)) = l(t_i(q))$ for all $q \in Q_2$, all the edges $e_1 \dots e_{i-1} \frac{1}{2} e_{i+1} \dots e_{|E|}$, with $e_j \in \{0, 1\}$ for $j \neq i$, have the same label. Denote this as the label l_i . The label of a general cell $q \in Q_n$ is the multiset of n labels $l_{j_1} \dots l_{j_n}$ where the j 's are exactly those indexes in the representation of q for which e_j has value $\frac{1}{2}$.*

As is the case with multi-modal logics or propositional dynamic logics [26], we extend *HDML* to have a multitude of modalities indexed by some alphabet Σ (the alphabet of the *HDA* in our case). This will be the same alphabet as that of the Mazurkiewicz trace represented by the *HDA*. In propositional dynamic logic there is an infinite number of modalities because they are indexed by an alphabet consisting of the regular expressions; yet these can be expressed in terms of a finite number of basic modalities (indexed by only the basic expressions). In our case we consider only an unstructured alphabet Σ which is considered finite.

Definition 3.14 (labeled modalities). *Consider two labeled modalities during $\{a\}\phi$ and after $\langle a \rangle\phi$ where $a \in \Sigma$ is a label from a fixed alphabet. The interpretation of the labeled modalities is given as:*

$$\begin{aligned} \mathcal{H}, q \models \{a\}\phi & \text{ iff assuming } q \in Q_n \text{ for some } n, \exists q' \in Q_{n+1} \text{ s.t.} \\ & s_i(q') = q \text{ for some } 1 \leq i \leq n, l(q') = l(q)a \text{ and } \mathcal{H}, q' \models \phi. \\ \mathcal{H}, q \models \langle a \rangle\phi & \text{ iff assuming } q \in Q_n \text{ for some } n, \exists q' \in Q_{n-1} \text{ s.t.} \\ & t_i(q) = q' \text{ for some } 1 \leq i \leq n, l(q) = l(q')a \text{ and } \mathcal{H}, q' \models \phi. \end{aligned}$$

Having the labeled modalities one can get the unlabeled variants as a disjunction over all labels

$$\{\}\phi \triangleq \bigvee_{a \in \Sigma} \{a\}\phi$$

The same as in Proposition 3.2 we captured axiomatically in the basic *HDML* language the Kripke models, the question now is whether we can capture in the basic *HDML* language with labeled modalities the Mazurkiewicz traces. The initial results in Lemma 3.15 cast the restrictions on labeled event structures of Proposition 3.12 into the *HDA* setting in the view discussed above. Nevertheless, the context-independence property of the labeling function λ is special and we discuss it afterwards.

Lemma 3.15 (trace restrictions in *HDA*).

*The notion of empty conflict relation from Definition 3.10 is captured in *HDML* by the axiom:*

$$a \neq b \models (\{a\}^\top \wedge \{b\}^\top) \rightarrow (\{a\}\{b\}^\top \wedge \{b\}\{a\}^\top) \quad (20)$$

⁵We remove also all those cells of higher dimension that are connected with the 0-dimensional cells that we have removed.

The notion of nice labeling from Proposition 3.12 is captured in HDML by the axiom:

$$\models \langle a \rangle \top \rightarrow \neg \{a\} \top \quad (21)$$

The notion of dependent actions a and b from Definition 3.9 is captured in HDML by the axiom:

$$\models \langle a \rangle \top \rightarrow \neg \{b\} \top \quad (22)$$

Proof. Mazurkiewicz traces do not employ the notion of conflict relation of the event structures. In other words, traces are encoded as event structures with an empty conflict relation. To such event structures the two restrictions of Definition 3.10 do not apply, being vacuously satisfied. Therefore, the Mazurkiewicz traces become, in this view, just configuration structures with the labeling function restricted as in Proposition 3.12. Because the conflict relation is what captures choices in event structures and in higher dimensional automata, the Mazurkiewicz traces are just linear models, unable to capture choices.

The axiom (20) restricts *HDA*s to not have choices. Essentially the axiom says that if in some cell one can start two different events (with different labels) then these two events are concurrent, i.e., the two during modalities can be stacked one on top of the other. Note that the axiom talks only about different labels. Choices between events with the same label are still allowed. To remove this form of nondeterminism we just need to add the modal axiom for determinism: $\models \{a\} \varphi \rightarrow \llbracket a \rrbracket \varphi$.

Such restricted *HDA*s still allow for *autoconcurrency* which is not the case in Mazurkiewicz traces. The *nice labeling* axiom (21) removes autoconcurrency. It basically says that two events with the same label cannot be concurrent; i.e., if an event labeled with a has been started then no other event labeled with a can start. Note that this axiom is meaningful on transitions and cells of higher dimension, but not in states; i.e., it is meaningful during the execution of the already started a -labeled events, not before starting them.

The last axiom (22) models the fact that two dependent actions are not concurrent, which is the last restriction in the Definition 3.9 of Mazurkiewicz traces. Note that the nice labeling restriction says that the dependence relation is reflexive, as required in Definition 3.9. \square

We could not capture the context-independent restriction on the labeling because it does not have just a universal presentation, so that we can capture it with axioms. This restriction is existential in nature, looking through all the higher dimensional automaton for some particular events. In fact it has a mixture of existential and universal assertions. Precisely, a labeling being context-independent is as saying that: if there exists throughout the *HDA* two events labeled with a and b which are concurrent, then all the pairs of events from the same *HDA* that are labeled with a and b must be concurrent. Or we can characterize it otherwise with the notion of *not-concurrent* as: if there exists throughout the *HDA* two events labeled with a and b which are not concurrent, then all the pairs of events from the same *HDA* that are labeled with a and b must not be concurrent. We can also have another view on this property, using two validities: either all the pairs of events labeled with a and b are not concurrent (i.e., axiom (22)) or all the pairs of events labeled with a and b are concurrent.

We conjecture that the context-independent restriction on the labeling function cannot be captured just in the basic *HDML* language, but the more expressive temporal operators are needed, which can talk about the whole *HDA* structure in an existential manner. Maybe just the *eventually* temporal modality is enough, instead of the stronger *Until* operator. Yet another question is whether just the LTL-like *Until* operator from Definition 3.6 is enough.

In the remainder of this section we show how the LTrL logic of [9] and the ISTL logic of [18, 10] is captured in the higher dimensional framework. These logics, as well as those presented in [7, 24], are interpreted in some particular configuration of a Mazurkiewicz trace (or of a restricted partial order). We take the view of Mazurkiewicz traces as restricted labeled posets from Proposition 3.9 but we use their representation using their corresponding family of configurations as in Proposition 3.12. Therefore, we now interpret *HDML* over restricted *HDA*s as we discussed above.

Proposition 3.16 (encoding LTrL). *The language of LTrL consists of the propositional part of HDML together with the following two definitions:*

- of the *Until* operator $\varphi \overline{\mathcal{U}} \varphi' \triangleq (\varphi \vee \langle \rangle \top) \mathcal{U}^l (\varphi' \wedge \neg \langle \rangle \top);$
- and the next step operator, for $a \in \Sigma$, $\overline{\langle a \rangle} \varphi \triangleq \{a\} \langle a \rangle \varphi.$

When interpreted only in the states of a HDA representing a Mazurkiewicz trace this language has the same behavior as the one presented in [9]

Proof. The states of the HDA are the configurations of the Mazurkiewicz trace. Thus, our definition of the LTrL language is interpreted in one trace at one particular configuration; as is done in [9]. The original semantics of LTrL uses transitions from one configuration to another labeled by an element from the alphabet Σ of the trace. It is easy to see that our syntactic definition of $\overline{\langle a \rangle} \varphi$ has the same interpretation as the corresponding one in [9]. The proof is similar to the proof of Theorem 3.3. In particular, when $\overline{\langle a \rangle} \varphi$ is interpreted in some state of the HDA, i.e., in a configuration of the trace, then the formula φ must hold in the state reached by going through a transition labeled with a . This means that we just made a single step, cf. the definition of [9], from the initial configuration to a new one where one new event labeled by a has been added.

The *Until* operator of [9] has the same definition as the one in standard LTL but adapted to the Mazurkiewicz traces setting; thus, in the syntactic definition of $\overline{\mathcal{U}}$ we use the LTL-like \mathcal{U}^l from Definition 3.6. \square

The ISTL logic is interpreted over communicating sequential agents (CSA), which are a restricted form of partial orders that still allows choices (as opposed to Mazurkiewicz traces). ISTL interprets the CTL until operator in configurations of a CSA. Therefore, we first need to find the exact restriction of HDAs modeling CSA and then just use the syntactic definition $\exists \mathcal{U}$ of Proposition 3.7. We do not go into details here but discuss the undecidability results for $\exists \mathcal{U}$.

In [17] the $\exists \mathcal{U}$ is interpreted only over Mazurkiewicz traces and an undecidability proof is given using a simple trace that looks like a grid, with only two labels that are independent. The proof of [10] uses a simple CSA but which allows choices. Intuitively, [10] builds infinitely many grids as in [17]. Both these proofs work with infinite partial orders (i.e., infinitely many events): [17] works on an infinite grid; whereas [10] works with infinitely many finite grids. There are two stages in these algorithms: the first is to encode all and only these infinite structures with some formula (for which the *Until* definitions are not even needed, but only their weaker forms like $\exists G$ are enough); the second stage is to encode the actual tests in the undecidability problem (the tiling problem in [17] and the Post correspondence problem in [10]). The first stage can be seen as setting the board for the undecidable problem.

We do not pursue further here investigation into the (un)decidability of HDML with the *Until* operator.

4 Expressiveness in terms of bisimulations

There are various ways of investigating the expressiveness of a logic. One way that we explored in the previous section is to see what other logics can be syntactically encoded into the studied logic and to isolate the exact restriction of the studied logic (and its models) that belongs to the encoded logic.

Another way of looking at the expressiveness of a modal logic is by investigating the kind of bisimulation that it captures. In this section we do this for HDML, with the aim to get more insights into the distinguishing power of the basic language of HDML. By distinguishing power we mean what kind of (two) models can be distinguished by a single HDML formula and what models are indistinguishable. The notion of *indistinguishable* is given through an appropriate bisimulation; i.e., if the two models are bisimilar (for some specific notion of bisimulation) then an observer cannot distinguish them. The observer, in our case, has only the power to test logical HDML formulas on the two models. Since we will refer to works that consider labeled transition systems, we will use the labeled versions of the HDML modalities as in Definition 3.14.

Other expressiveness results for modal (temporal) logics include investigations into what exact subset of first (or second) order logic they capture, as is done for linear time temporal logic [27] (see [28] for an overview) or for the LTrL [9]. We do not pursue this line of research here.

HDML captures precisely the split-bisimulation and is strictly coarser than ST-bisimulation or history preserving bisimulation. Therefore, we confine our presentation here to only split-bisimulation, and discuss shortly the reasons that make *HDML* less expressive than the other bisimulations on *HDA*s.

Definition 4.1 (split-bisimulation). *The split of a finite path π in a HDA is the sequence $\text{split}(\pi) \triangleq \sigma_1 \dots \sigma_n$ where $\sigma_i = l_i(q^i)^+$ if $\alpha^i = s$ and $\sigma_i = l_i(q^i)^-$ if $\alpha^i = t$ for $1 \leq i \leq n$. Two higher dimensional automata (\mathcal{H}_A, q_A^0) and (\mathcal{H}_B, q_B^0) (with q_A^0 and q_B^0 two initial cells) are split-bisimulation equivalent if there exists a binary relation R between their paths starting at q_A^0 respectively q_B^0 that respects the following:*

1. if $\pi_A R \pi_B$ then $\text{split}(\pi_A) = \text{split}(\pi_B)$;
2. if $\pi_A R \pi_B$ and $\pi_A \rightarrow^* \pi'_A$ then $\exists \pi'_B$ with $\pi_B \rightarrow^* \pi'_B$ and $\pi'_A R \pi'_B$;
3. if $\pi_A R \pi_B$ and $\pi_B \rightarrow^* \pi'_B$ then $\exists \pi'_A$ with $\pi_A \rightarrow^* \pi'_A$ and $\pi'_A R \pi'_B$;

Denote this as $(\mathcal{H}_A, q_A^0) \approx_s (\mathcal{H}_B, q_B^0)$.

The ST-bisimulation replaces the first requirement with equality between ST-traces of the two paths. Intuitively, the ST-trace of a path is like the split-trace only that the end labels $l_i(\cdot)^-$ are keeping count of which start label they match with; i.e., $l_i(\cdot)^j$ where at the j^{th} point the corresponding event has been started. Therefore, ST-traces know exactly which event ends; whereas the split-traces may confuse this. History preserving bisimulation is defined using the notions of adjacency and homotopy for *HDA* and intuitively, for some cell in the *HDA* we have a grip on its history also. Thus, history preserving bisimulation has access to the whole partially ordered history of the current executing events, ST-bisimulation has access only to some point from the past (i.e., the origin of some event), whereas the split-bisimulation has only a notion of previous step on the path. We come back to these intuitions throughout this section.

A modal logic is said to capture some equivalence relation \sim if for any two models \mathcal{H} and \mathcal{H}' , they are equated by the \sim relation iff they are *modally equivalent*.

Definition 4.2 (modal equivalence). *Define the HDML modal equivalence as the relation $\overset{\text{HDML}}{\sim}$ s.t.:*

$$(\mathcal{H}, q) \overset{\text{HDML}}{\sim} (\mathcal{H}', q') \text{ iff } \forall \varphi : \mathcal{H}, q \models \varphi \Leftrightarrow \mathcal{H}', q' \models \varphi.$$

To keep the presentation simple we will work with frames instead of models; i.e., with no propositional constants. Before presenting the formal result note that *HDML* can distinguish branching points, as is the case with bisimulations opposed to trace equivalences; the standard example in process algebras $(a(b+c) \text{ vs. } ab+ac)$ is distinguished by the *HDML* formula $\llbracket a \rrbracket [a](\{b\} \top \wedge \{c\} \top)$. *HDML* also distinguishes between interleaving and split-2 concurrency, where the standard example of $a||b$ vs. $ab+ba$ is distinguished by the formula $\{a\}\{b\} \top$ which holds only for $a||b$.

Proposition 4.3 (*HDML* captures split-bisimulation).

The relations $\overset{\text{HDML}}{\sim}$ and \approx_s coincide.

Proof. Proving the inclusion $\approx_s \subseteq \overset{\text{HDML}}{\sim}$ is simple. Use induction on the structure of the formula and use the last two conditions for \approx_s with a smallest extension of the paths, i.e., when only one simple step is added to the path. The split-traces give the label and the s or t needed (when working with $\{\}$ respectively $\langle \rangle$).

Proving the other inclusion $\overset{\text{HDML}}{\sim} \subseteq \approx_s$ needs the standard assumptions of finite nondeterminism (or image-finite as it is also known) and finite concurrency. This proof uses *reductio ad absurdum* to show that the relation $\overset{\text{HDML}}{\sim}$ is respecting the three conditions of Definition 4.1. Showing these conditions for all the paths is inductive, starting

with the empty path and making only simple steps of extending the paths in the conditions 2 and 3, because this is enough to get the general form of these conditions.

For the empty paths the condition 1 is trivially satisfied. We work here with simple steps that extend the path with s maps labeled by some a ; and the other map t is treated analogous. Consider the initial cells $q_A \stackrel{HDML}{\sim} q_B$, and that $s_i(q_A^1) = q_A$ labeled by a (i.e., we extend the empty split-trace with a^+). We will assume that there is no way of extending (with a single step) the empty path in \mathcal{H}_B cf. condition 2 of Definition 4.1: i.e., $\nexists q_B^1$ s.t. $s_i(q_B^1) = q_B$, for some i , and labeled with a , and modal equivalent $q_B^1 \stackrel{HDML}{\sim} q_A^1$. If the assumption holds because there is no way of starting an a -labeled event then the modal formula $\llbracket a \rrbracket \perp$ holds in q_B . But because in q_A holds $\{a\} \top$ and $q_A \stackrel{HDML}{\sim} q_B$ then we get a contradiction because $q_B \models \{a\} \top \wedge \llbracket a \rrbracket \perp$. Because of the finite nondeterminism and finite concurrency, the set of cells $\{q_B^1, \dots, q_B^n\}$ reachable by an s map labeled by a from q_B , is finite. It remains to check the modal equivalence of the new cells. Clearly the split-traces of the new paths are the same because we extend with the same s map labeled with the same a . Assume that for each cell q_B^i there exists some formula φ^i that holds in q_A^1 but not in q_B^i . Hence, $q_A \models \{a\}(\varphi^1 \wedge \dots \wedge \varphi^n)$ but $q_B \not\models \{a\}(\varphi^1 \wedge \dots \wedge \varphi^n)$, which is a contradiction with the fact that q_A and q_B are modal equivalent (i.e., model the same formulas). \square

Because split-bisimulation can distinguish choices, then *HDML* can distinguish all the examples of [29] that were meant there to distinguish between the many trace-based equivalences. In particular, *HDML* distinguishes the X_n^{odd} and X_n^{even} pomset processes (in their *HDA* representation) which are meant to distinguish the split- $n+1$ from the split- n trace equivalences (e.g., the formula $\{1\}(\{2\} \top \wedge \langle 1 \rangle (\{0\} \langle 0 \rangle \{2\} \langle 2 \rangle \{2\} \top \wedge \llbracket 2 \rrbracket \llbracket 2 \rrbracket \llbracket 0 \rrbracket \llbracket 0 \rrbracket \neg \{1\} \top))$ distinguishes the two examples in [29, Figure 2] because it holds on X_2^{even} but not on X_2^{odd}). Also, *HDML* can distinguish the examples in [29, Figure 3] because the formula $\llbracket a \rrbracket \llbracket b \rrbracket [b][a]\{c\} \top$ holds in the pomset process Y but not in X (in their *HDA* presentation). This example is meant in [29] to distinguish the ST-trace equivalence from all the split- n trace equivalences because the two pomset processes are indistinguishable by any of the split- n trace equivalences.

Nevertheless, when it comes to bisimulation equivalences *HDML* captures only split-bisimulation. Intuitively, the examples above can be distinguished by *HDML* because they have different branching points before the problematic autoconcurrency square. *HDML* becomes stuck when it has to deal with autoconcurrency; i.e., when in a concurrency square with both sides labeled the same, *HDML* cannot distinguish which of the two events it finishes. But ST-bisimulation and history preserving bisimulation can distinguish the two events by looking at the history. In particular, *HDML* is unable to distinguish any of the “owl” examples of [29] which are meant to separate the split- n -bisimulations.

In conclusion, *HDML* sits pretty low in the equivalences spectrum of van Glabbeek and Vaandrager [29], capturing only split-bisimulation. An interesting question for future work is what is a minimal extension to *HDML* that captures ST-bisimulation, or history preserving bisimulation?

5 Conclusion

We have investigated a modal logic called *HDML* which is interpreted over higher dimensional automata. The language of *HDML* is simple, capturing both the notions of “during” and “after”. The associated semantics is intuitive, accounting for the special geometry of the *HDAs*. An adaptation of the filtration method was needed to prove decidability. We have associated to *HDML* an axiomatic system which incorporates the standard modal axioms and has a few natural axioms extra, which are related to the cubical laws and to the dimensions of *HDAs*.

We isolated axiomatically the class of *HDAs* that encode Kripke structures and shown how standard modal logic is encoded into *HDML* when interpreted only over these restricted *HDAs*. We then showed how to extend the expressiveness of *HDML* using the *Until* operator by defining two kinds of *Until* over *HDAs*: one \mathcal{U}^l in the LTL style and one \mathcal{U}^c in the CTL style. Using the \mathcal{U}^c we showed how to encode syntactically the CTL $\exists \mathcal{U}$ into *HDML* when interpreted over the Kripke *HDAs*. We also showed how weaker concurrency models like

Mazurkiewicz traces or (restrictions of) event structures can be encoded in *HDML* and how some of their specific properties can be captured axiomatically only in the basic language of *HDML*. We also looked at encoding specific logics for these restricted models (particularly the LTrL and ISTL) in the extensions of *HDML* with the *Until* operators.

In the last technical section we investigated the distinguishing power of *HDML* and isolated the basic language of *HDML* as capturing exactly the split-bisimulation. Nevertheless, the power to distinguish different branching points allowed *HDML* to distinguish all the examples of [29] that were meant there to separate the split-n-trace equivalences and the ST-trace equivalence. In this respect we gave some discussions trying to identify the weak points of *HDML* compared to ST-bisimulation or history preserving bisimulation.

Interesting further work is to look more into the relation of *HDML* (and its temporal extensions) with other logics for weaker models of concurrency like with the modal logic of [11] for event structures or other logics for Mazurkiewicz traces. Particularly interesting is to give details of how or if the undecidability results of [10, 18] are applicable to our setting.

When investigating deeper the extensions of *HDML* wrt. the captured bisimulations, the work of [30] is of particular relevance and comparisons with the logics presented there worth wild.

Acknowledgements: I would like to thank Martin Steffen and Olaf Owe for their useful comments, as well as to the anonymous reviewers of previous drafts of this work.

References

- [1] C. Prisacariu, Modal Logic over Higher Dimensional Automata, in: P. Gastin, F. Laroussinie (Eds.), 21st International Conference on Concurrency Theory (CONCUR10), Vol. 6269 of LNCS, Springer, 2010, pp. 494–508.
- [2] V. R. Pratt, Modeling concurrency with geometry, in: Principles of Programming Languages (POPL’91), 1991, pp. 311–322.
- [3] R. J. van Glabbeek, On the Expressiveness of Higher Dimensional Automata, Theoretical Computer Science 356 (3) (2006) 265–290.
- [4] R. J. van Glabbeek, U. Goltz, Refinement of actions and equivalence notions for concurrent systems, Acta Informatica 37 (4/5) (2001) 229–327.
- [5] V. Gupta, Chu Spaces: A Model of Concurrency, Ph.D. thesis, Stanford University (1994).
- [6] V. R. Pratt, Transition and Cancellation in Concurrency and Branching Time, Mathematical Structures in Computer Science 13 (4) (2003) 485–529.
- [7] M. Mukund, P. S. Thiagarajan, Linear Time Temporal Logics over Mazurkiewicz Traces, in: Mathematical Foundations of Computer Science (MFCS’96), Vol. 1113 of LNCS, Springer, 1996, pp. 62–92.
- [8] V. Diekert, P. Gastin, From local to global temporal logics over Mazurkiewicz traces, Theoretical Computer Science 356 (1-2) (2006) 126–135.
- [9] P. S. Thiagarajan, I. Walukiewicz, An Expressively Complete Linear Time Temporal Logic for Mazurkiewicz Traces, Information and Computation 179 (2) (2002) 230–249.
- [10] R. Alur, D. Peled, Undecidability of partial order logics, Information Processing Letters 69 (3) (1999) 137–143.
- [11] K. Lodaya, M. Mukund, R. Ramanujam, P. S. Thiagarajan, Models and Logics for True Concurrency, Tech. Rep. IMSc-90-12, Inst. Mathematical Science, Madras, India (1990).
- [12] K. Lodaya, R. Parikh, R. Ramanujam, P. S. Thiagarajan, A Logical Study of Distributed Transition Systems, Information and Computation 119 (1) (1995) 91–118.
- [13] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite state concurrent systems using temporal logic specifications, in: Principles of Programming Languages (POPL’83), 1983, pp. 117–126.
- [14] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Vol. 53 of Cambridge Tracts in Theoretical Computer Science, Cambridge Univ. Press, 2001.

- [15] V. R. Pratt, A Practical Decision Method for Propositional Dynamic Logic: Preliminary Report, in: Symposium on Theory of Computing (STOC'78), ACM Press, 1978, pp. 326–337.
- [16] A. Pnueli, The temporal logic of programs, in: Symposium on Foundations of Computer Science (FOCS'77), IEEE Computer Society Press, 1977, pp. 46–57.
- [17] W. Penczek, On Undecidability of Propositional Temporal Logics on Trace Systems, *Information Processing Letters* 43 (3) (1992) 147–153.
- [18] R. Alur, K. L. McMillan, D. Peled, Deciding Global Partial-Order Properties, *Formal Methods in System Design* 26 (1) (2005) 7–25.
- [19] A. W. Mazurkiewicz, Basic notions of trace theory., in: REX Workshop, Vol. 354 of LNCS, Springer, 1988, pp. 285–363.
- [20] V. R. Pratt, Modeling Concurrency with Partial Orders, *Journal of Parallel Programming* 15 (1) (1986) 33–71.
- [21] M. Nielsen, G. D. Plotkin, G. Winskel, Petri nets, event structures and domains., in: *Semantics of Concurrent Computation*, Vol. 70 of LNCS, Springer, 1979, pp. 266–284.
- [22] V. R. Pratt, Higher dimensional automata revisited, *Mathematical Structures in Computer Science* 10 (4) (2000) 525–548.
- [23] B. Rozoy, P. S. Thiagarajan, Event structures and trace monoids, *Theoretical Computer Science* 91 (2) (1991) 285–313.
- [24] V. Diekert, P. Gastin, LTL Is Expressively Complete for Mazurkiewicz Traces, in: *International Colloquium on Automata, Languages and Programming (ICALP'00)*, Vol. 1853 of LNCS, Springer, 2000, pp. 211–222.
- [25] K. Lodaya, R. Ramanujam, P. S. Thiagarajan, Temporal Logics for Communicating Sequential Agents: I, *International Journal on Foundations of Computer Science* 3 (2) (1992) 117–159.
- [26] D. Harel, D. Kozen, J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.
- [27] H. Kamp, *Tense Logic and the Theory of Linear Orders*, Ph.D. thesis, UCLA (1968).
- [28] E. A. Emerson, Temporal and Modal Logic, in: *Handbook of Theoretical Computer Science*, Volume B, 1990, pp. 995–1072.
- [29] R. J. van Glabbeek, F. W. Vaandrager, The Difference between Splitting in n and $n+1$, *Information and Computation* 136 (2) (1997) 109–142.
- [30] P. Baldan, S. Crafa, A logic for true concurrency, in: P. Gastin, F. Laroussinie (Eds.), *21st International Conference on Concurrency Theory (CONCUR10)*, Vol. 6269 of LNCS, Springer, 2010, pp. 147–161.

A Completeness

This section is not finished. It presents the main ideas and steps needed to prove the completeness of the axiomatic system for *HDML* from Table 2; but still some details need to be fit into place. Any comments on this proof are welcome. The complications and details of this completeness proof are as such because of the intricate geometrical structure of the *HDA* model that we work with. Some of the inductive reasoning that is needed does not follow standard patterns, and makes arguments more involved.

We first fix some terminology and notation. Because of the finite model property for *HDML* from Theorem 2.13 and because compactness fails cf. Theorem 2.17, we are inclined to use atoms in the proof of completeness for *HDML* and build finite canonical models (instead of using maximal consistent sets and standard canonical model).

Definition A.1 (atoms). Recall from Definition 2.4 that $\mathcal{C}(\varphi)$ is the subformula closure set of some given formula φ . Denote by $\neg\mathcal{C}(\varphi) = \mathcal{C}(\varphi) \cup \{\neg\varphi' \mid \varphi' \in \mathcal{C}(\varphi)\}$ the set of subformulas together with their negated forms. A set of formulas A is called an atom for φ if A is a maximal consistent subset of $\neg\mathcal{C}(\varphi)$. Denote $At(\varphi)$ the set of all atoms for φ . For an atom $A \in At(\varphi)$ denote by \hat{A} the formula obtained as $\phi_1 \wedge \dots \wedge \phi_n$ with $\phi_i \in A$.

Intuitively, atoms are sets of formulas which are free of immediate propositional inconsistencies (like $\phi \wedge \neg\phi$).

Lemma A.2 (properties on atoms). Standard results for atoms tell us that for some formula φ and any atom $A \in At(\varphi)$ is the case that:

- (i). for all $\psi \in \neg\mathcal{C}(\varphi)$ then only one of ψ or $\neg\psi$ are in A ;
- (ii). for all $\psi \rightarrow \psi' \in \neg\mathcal{C}(\varphi)$ then $\psi \rightarrow \psi' \in A$ iff whenever $\psi \in A$ then $\psi' \in A$;
- (iii). if $\psi \in \neg\mathcal{C}(\varphi)$ and ψ is consistent then there exists an $A \in At(\varphi)$ s.t. $\psi \in A$; (This is an analog of Lindenbaum's Lemma.)
- (iv). any consistent set of formulas $S \subseteq \neg\mathcal{C}(\varphi)$ can be grown to an atom $A_S \in At(\varphi)$.

Definition A.3 (canonical saturated HDA). A HDA is called canonical for the formula φ if a canonical labeling $\lambda : Q \rightarrow At(\varphi)$ can be attached to the HDA. A labeling function is canonical if the following conditions hold:

1. for any $q_n \in Q_n, q_{n-1} \in Q_{n-1}$, with $n > 0$, and $\forall 0 \leq i \leq n$, if $s_i(q_n) = q_{n-1}$ then $\forall \psi \in \neg\mathcal{C}(\varphi)$ if $\llbracket \psi \rrbracket \in \lambda(q_{n-1})$ then $\psi \in \lambda(q_n)$,
2. for any $q_n \in Q_n, q_{n-1} \in Q_{n-1}$, with $n > 0$, and $\forall 0 \leq i \leq n$, if $t_i(q_n) = q_{n-1}$ then $\forall \psi \in \neg\mathcal{C}(\varphi)$ if $\llbracket \psi \rrbracket \in \lambda(q_n)$ then $\psi \in \lambda(q_{n-1})$.

A canonical HDA is called saturated if:

1. whenever $\{\} \psi \in \lambda(q_{n-1})$ then $\exists q_n \in Q_n$ and $\exists 0 \leq i \leq n$ s.t. $s_i(q_n) = q_{n-1}$ and $\psi \in \lambda(q_n)$,
2. whenever $\langle \rangle \psi \in \lambda(q_n)$ then $\exists q_{n-1} \in Q_{n-1}$ and $\exists 0 \leq i \leq n$ s.t. $t_i(q_n) = q_{n-1}$ and $\psi \in \lambda(q_{n-1})$.

Lemma A.4 (truth lemma). In a canonical saturated HDA \mathcal{H} for a formula φ , with the valuation defined as $\mathcal{V}(q_n) = \{\phi \in \Phi_B \mid \phi \in \lambda(q_n)\}$, it holds that $\mathcal{H}, q_n \models \psi$ iff $\psi \in \lambda(q_n)$, for any $\psi \in \neg\mathcal{C}(\varphi)$.

Proof. By induction on the structure of ψ .

Base case: $\psi = \phi \in \Phi_B$. From the definition we have $\mathcal{H}, q_n \models \phi$ iff $\phi \in \mathcal{V}(q_n)$ iff $\phi \in \lambda(q_n)$.

Inductive step: The case for the Boolean connectives follows easily from the properties on atoms. Finally we treat cases for the two modalities. Consider the during modality. The left to right direction is based on the canonicity of \mathcal{H} .

We prove $\mathcal{H}, q_n \models \{\} \varphi \Rightarrow \{\} \varphi \in \lambda(q_n)$. From the definition we know that $\exists q' \in Q_{n+1}$ and $\exists 0 \leq i \leq n+1$ s.t. $s_i(q') = q_n$ and $\mathcal{H}, q' \models \varphi$. From the induction hypothesis we have that $\mathcal{H}, q' \models \varphi$ iff $\varphi \in \lambda(q')$. Together with the canonicity of \mathcal{H} we have that $\{\} \varphi \in \lambda(q_n)$. Proof finished.

For the right to left direction we use the saturation of \mathcal{H} . We prove $\{\} \varphi \in \lambda(q_n) \Rightarrow \mathcal{H}, q_n \models \{\} \varphi$. Using the saturation of \mathcal{H} we have that $\exists q_{n+1} \in Q_{n+1}$ and $\exists 0 \leq i \leq n+1$ s.t. $s_i(q_{n+1}) = q_n$ and $\varphi \in \lambda(q_{n+1})$. By the induction hypothesis it implies that $\mathcal{H}, q_{n+1} \models \varphi$. Thus, by the definition we have that $\mathcal{H}, q_n \models \{\} \varphi$.

The proof for the $\langle \rangle$ modality is symmetric using the second conditions of canonicity and saturation of \mathcal{H} . \square

For modal logics over complex structures like *HDA*s the step-by-step method of proving completeness is a first candidate. But we cannot use it in the standard way with maximal consistent sets as the cells of the *HDA*. Instead we will use atoms, i.e., finite maximal consistent sets. On the other hand, the standard way of using atoms in completeness proofs is to build a finite canonical model and show that it respects the required special structure. This is not easy in our case. In consequence we use a step-by-step method for building a finite model and label the cells with atoms. This model will have the right *HDA* structure and will respect canonicity properties required by a truth lemma. In this sense we adapt and combine the two methods of step-by-step and atoms-based finite canonical models to *HDML*. On the other hand the main difficulty of our proof is the construction method which is rather involved. Note that we prove a weak completeness result, which is normal because a strong completeness is out of reach because of the compactness failure.

A first attempt to prove completeness is to try to build a canonical saturated model for any consistent formula. This fails, partly because *HDML* is a forward looking logic but the special cubical geometry of the *HDA*s require to construct the backwards part of the *HDA* (that which is not reachable through the two modalities of *HDML*). But it is not possible to guarantee the canonicity for this part. Nevertheless, the following notions and results tell us that we can ignore canonicity for this part of the model. Therefore when doing the actual step-by-step construction of the required *HDA* for some arbitrary consistent formula we will concentrate on respecting canonicity only for the relevant (cf. the results below) part of the structure.

Definition A.5 (pseudo *HDA*). *For a HDA \mathcal{H} and a cell $q \in \mathcal{H}$ we call the forward generated pseudo *HDA* for the cell q (or pseudo *HDA* for short) the structure $\mathcal{H}_q^p = (Q', \bar{s}', \bar{t}', l')$ obtained from \mathcal{H} by the following generative definition:*

- $q \in Q'$;
- if $q \in Q'$ then $\forall q_s \in Q$ if it exists i s.t. $s_i(q_s) = q$ then $q_s \in Q'$;
- if $q \in Q'$ then $\forall q_t \in Q$ if it exists i s.t. $t_i(q) = q_t$ then $q_t \in Q'$;
- no other cell of Q is in Q' ;

and where $\bar{s}' \triangleq \bar{s}|_{Q'}$, $\bar{t}' \triangleq \bar{t}|_{Q'}$, and $l' \triangleq l|_{Q'}$ are the respective restriction to this new set of cells Q' .

Intuitively, the pseudo *HDA*s are similar to the idea of a point-generated submodel in standard modal logic. The following lemma intuitively says that *HDML* satisfaction is invariant under pseudo model construction.

Lemma A.6 (invariance under pseudo *HDA*s). *For a HDA \mathcal{H} and a cell $q \in \mathcal{H}$ for which \mathcal{H}_q^p denotes the pseudo *HDA* for q , then for any *HDML* formula φ and any cell $q^p \in \mathcal{H}_q^p$, we have:*

$$\mathcal{H}, q^p \models \varphi \text{ iff } \mathcal{H}_q^p, q^p \models \varphi.$$

Proof. The proof uses induction on the structure of the formula φ . Since the pseudo *HDA* does not change the valuation then the base case for propositional constants and the inductive cases for the Boolean operators are trivial, as we have to look at the same cell.

It remains to treat the inductive cases for the two *HDML* modalities; we will treat only $\varphi = \{\}\psi$, and the other modality is treated analogous. Since the set of cells of the pseudo *HDA* is just a subset of the original \mathcal{H} , i.e., $Q' \subseteq Q$, then the right-to-left implication is immediate (using induction on the subformula ψ). For the left-to-right implication consider that $\mathcal{H}, q^p \models \varphi$ and we show that $\mathcal{H}_q^p, q^p \models \varphi$. From the semantic definition we have that it exists $s_i(q_{n+1}) = q^p$, for some i , with $\mathcal{H}, q_{n+1} \models \psi$. From the pseudo *HDA* Definition A.5, since $q^p \in \mathcal{H}_q^p$ then also $q_{n+1} \in \mathcal{H}_q^p$ and $s'_i(q_{n+1}) = q^p$. From $\mathcal{H}, q_{n+1} \models \psi$, by the induction hypothesis we have that $\mathcal{H}_q^p, q_{n+1} \models \psi$. These imply the desired result $\mathcal{H}_q^p, q^p \models \{\}\psi$. \square

Definition A.7 (pseudo canonicity). *We call a HDA pseudo canonical for q if its pseudo *HDA* for q is canonical (cf. Definition A.3). A pseudo canonical *HDA* is called saturated if its pseudo *HDA* is saturated.*

Lemma A.8 (truth lemma for pseudo canonical HDAs). *In a HDA \mathcal{H} which is pseudo canonical for q and saturated wrt. a formula ϕ , with the valuation defined as $\mathcal{V}(q_n) = \{\phi \in \Phi_B \mid \phi \in \lambda(q_n)\}$, then for any $\psi \in \neg\mathcal{C}(\phi)$ and $q_n \in \mathcal{H}_q^p$ it holds that*

$$\mathcal{H}, q_n \models \psi \text{ iff } \psi \in \lambda(q_n).$$

Proof. The proof follows from the Truth Lemma A.4. □

To prove completeness of the axiomatic system all that remains is to show that for any consistent formula ϕ we can build such a pseudo canonical saturated HDA. During the step-by-step construction process we constantly struggle to saturate the HDA (that we work with) while respecting the pseudo canonicity. Such not saturated HDAs are called *defective*, as they may have defects, which we formally define below. But important is that any of these defects can be repaired. This is what the repair lemma does, using the two *enriching* and *lifting* constructions. The completeness theorem then shows that while starting with a minimal pseudo canonical HDA we can incrementally build a defect free pseudo canonical HDA, i.e., a pseudo canonical and saturated HDA.

Definition A.9 (defects). *There are two types of defects for \mathcal{H} (corresponding to a violation of a saturation condition):*

- a D1 defect of \mathcal{H} is a cell $q_n \in Q_n$ with $\{\} \psi \in \lambda(q_n)$ for which there is no $q_{n+1} \in Q_{n+1}$ and no $1 \leq i \leq n+1$, with $s_i(q_{n+1}) = q_n$ and $\psi \in \lambda(q_{n+1})$;
- a D2 defect of \mathcal{H} is a cell $q_n \in Q_n$ with $\langle \rangle \psi \in \lambda(q_n)$ for which there is no $q_{n-1} \in Q_{n-1}$ and no $1 \leq i \leq n-1$, with $t_i(q_n) = q_{n-1}$ and $\psi \in \lambda(q_{n-1})$.

During the construction of the model we cannot work with atoms directly because we will revisit the same cell several times, each time needing to add more restrictions to its label. We are still working with atoms as labels, only that we consider all possible atoms that respect such properties (eg., all the atoms that could extend some consistent set of formulas). In the end of the construction we just pick one, to obtain the pseudo canonical saturated model we are seeking. The key result here is that all the constraints that are gathered during the construction should allow for the existence of at least one respecting atom. We use the following definitions.

Definition A.10 (potential labeling function). *We define a potential labeling function $\tilde{\lambda} : Q \rightarrow 2^{\mathbf{C}}$ which for any cell $q \in Q$ returns a set of constraints from \mathbf{C} . A constraint $c \in \mathbf{C}$ can be either a consistent set of formulas $S \in \mathcal{C}(\phi)$ (call this a set constraint) or a formula $\{\} \hat{A}$ or $\langle \rangle \hat{A}$, with $A \in \text{At}(\phi)$ an atom, (call these existential constraints). A potential labeling is called potential canonical iff there exists some labeling function λ s.t. for any cell $q \in Q$, $\lambda(q)$ is consistent with all the constraints $\tilde{\lambda}(q)$.*

Lemma A.11. *A potential labeling is not canonical iff any of the following is the case:*

- for some cell q the union of all the set constraints in $\tilde{\lambda}(q)$ is inconsistent;
- for some cell q there exists a formula $\phi \in A$ with A appearing in one of the existential constraints as $\{\} \hat{A} \in \tilde{\lambda}(q)$ (or as $\langle \rangle \hat{A} \in \tilde{\lambda}(q)$) for which there exists a corresponding formula $\llbracket \rrbracket \neg \phi$ (respectively $\llbracket \rrbracket \neg \phi$) in one of the set constraints of $\tilde{\lambda}(q)$.

Proof. □

Definition A.12. *For two HDAs, \mathcal{H}_1 and \mathcal{H}_2 , with corresponding potential canonical labellings $\tilde{\lambda}^1$ respectively $\tilde{\lambda}^2$, we say that \mathcal{H}_2 extends \mathcal{H}_1 (written $\mathcal{H}_2 \triangleright \mathcal{H}_1$) iff \mathcal{H}_2 has all the cells and maps of \mathcal{H}_1 and possibly some new cells and maps (i.e., some extra structure), and for all the old cells $q \in \mathcal{H}_1$ the constraints may only increase, i.e., $\tilde{\lambda}^1(q) \subseteq \tilde{\lambda}^2(q)$.*

Note that increasing the number of constraints means that there is less uncertainty about the ultimate atom that is going to label a cell.

The two constructions that we give below are working on pseudo canonical *HDA*s, where the minimal such *HDA* contains only one cell; this is the starting pseudo canonical *HDA* in the completeness Theorem A.24.

For a D1 defect, i.e., a cell q as in Definition A.9, the *enriching construction* adds one new cell that has q as one of its sources and is labeled with an atom containing ϕ . Moreover, all the other maps of this new cell need to be added, together with all the necessary new cells, respecting the cubical laws. The new enriched *HDA* will not have q as a D1 defect any more.

Lemma A.13 (enriching construction). *For an \mathcal{H} with an associated potential canonical labeling $\tilde{\lambda}$, and for a defective cell q (i.e., $\{\phi \in \tilde{\lambda}(q)\}$) the following construction, which we call enriching of the \mathcal{H} wrt. q and $\{\phi\}$ builds an \mathcal{H}' which extends \mathcal{H} (i.e., $\mathcal{H}' \triangleright \mathcal{H}$) and does not contain the defect of q nor introduces new defects for q .*

```

1 function enrich(n,q,φ){
2   Qn+1 := Qn+1 ∪ {qn+1}; // fresh cell
3   update map sn+1 s.t. sn+1(qn+1) = q;
4   add constraints  $\tilde{\lambda}(q_{n+1}) = \tilde{\lambda}(q_{n+1}) \cup \{\phi\} \cup \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(q)\}$ ;
5   add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \{\lambda(\widehat{q_{n+1}})\}$ ;
6   addSourceMaps(n+1,qn+1,0,0);
7   addTargetMaps(n+1,qn+1,0,0);
8 }
9 function addSourceMaps(k,q,m,q'){
10  if (k >= 1){
11    Qk-1 := Qk-1 ∪ {qk-11, ..., qk-1k-m}; // fresh cells
12    for (l=1 to m){
13      update map sk-l s.t. sk-l(q) = sk-m(sk-l+1(q'));
14      add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(s_{k-m}(s_{k-l+1}(q')))\}$ ;
15      add constraints  $\tilde{\lambda}(s_{k-l}(q)) = \tilde{\lambda}(s_{k-l}(q)) \cup \{\lambda(\widehat{q})\}$ ;
16    }
17    for (i=k-l-m to 1){
18      update map si s.t. si(q) = qk-1i;
19      update map sk-1 s.t. sk-1(qk-1i) = si(sk(q));
20      add constraints  $\tilde{\lambda}(q_{k-1}^i) = \tilde{\lambda}(q_{k-1}^i) \cup \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(s_i(s_k(q)))\} \cup \{\lambda(\widehat{q})\}$ ;
21      add constraints  $\tilde{\lambda}(s_{k-1}(q_{k-1}^i)) = \tilde{\lambda}(s_{k-1}(q_{k-1}^i)) \cup \{\lambda(\widehat{q_{k-1}^i})\}$ ;
22      addSourceMaps(k-1,qk-1i,k-1-m-i,q);
23      addTargetMaps(k-1,qk-1i,0,0);
24      add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(q_{k-1}^i)\}$ ;
25    }
26  }
27  function addTargetMaps(k,q,m,q'){
28    if (k >= 1){
29      Qk-1 := Qk-1 ∪ {qk-11, ..., qk-1k-m}; // fresh cells
30      for (l=0 to m-1){
31        update map tk-l s.t. tk-l(q) = tk+1-m(tk-l+1(q'));
32        add constraints  $\tilde{\lambda}(t_{k-l}(q)) = \tilde{\lambda}(t_{k-l}(q)) \cup \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(q)\}$ ;
33        add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \langle \lambda(\widehat{t_{k-l}(q)}) \rangle$ ;
34      }
35      for (i=k-m to 1){
36        update map ti s.t. ti(q) = qk-1i;
37        add constraints  $\tilde{\lambda}(q_{k-1}^i) = \{\psi \mid \llbracket \psi \rrbracket \in \tilde{\lambda}(q)\}$ ;
38        add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \langle \lambda(\widehat{q_{k-1}^i}) \rangle$ ;
39      }
40      if (k > 1){
41        for (j=1 to k-1){ // add k-1 maps sj to qk-1i cf. cubical laws

```

```

40   if (j < i) {
41     update map  $s_j$  s.t.  $s_j(t_i(q)) = t_{i-1}(s_j(q))$ ;
42     add constraints  $\tilde{\lambda}(q_{k-1}^i) = \tilde{\lambda}(q_{k-1}^i) \cup \{\psi \mid \llbracket \psi \rrbracket \psi \in \tilde{\lambda}(t_{i-1}(s_j(q)))\}$ ;
43     add constraints  $\tilde{\lambda}(t_{i-1}(s_j(q))) = \tilde{\lambda}(t_{i-1}(s_j(q))) \cup \{\lambda(q_{k-1}^i)\}$ ;
44   } else {
45     update map  $s_j$  s.t.  $t_i(s_{j+1}(q)) = s_j(t_i(q))$ ;
46     add constraints  $\tilde{\lambda}(q_{k-1}^i) = \tilde{\lambda}(q_{k-1}^i) \cup \{\psi \mid \llbracket \psi \rrbracket \psi \in \tilde{\lambda}(t_i(s_{j+1}(q)))\}$ ;
47     add constraints  $\tilde{\lambda}(t_i(s_{j+1}(q))) = \tilde{\lambda}(t_i(s_{j+1}(q))) \cup \{\lambda(q_{k-1}^i)\}$ ;
48   }
49   addTargetMaps (k-1,  $q_{k-1}^i$ , k-m-i, q);
50 }

```

Proof. The proof has five stages. The first three are meant to show that the enriched model is an extension of the old model (i.e., $\mathcal{H}' \triangleright \mathcal{H}$): 1) we first show that the structure of the old *HDA* is untouched, i.e., all old cells and maps are in place; 2) we then show that all set constraints are still consistent sets; 3) the third step shows that . Basically the steps two and three are corresponding to Lemma A.11 to show that the new potential labeling is still potential canonical, i.e., that there still exists a way of instantiating the constraints to atoms. The fourth stage shows that \mathcal{H}' is a model indeed, i.e., that all the maps are in place and all necessary cubical laws are respected. The last stage shows that the enriched model does not have the old defect and that no new defects are introduced in the potential labeling of the initial cell q .

First remark that we do not change the initial shape of the original \mathcal{H} ; we only add fresh cells and fresh maps for these cells; we also add maps to old cells connected to new cells. This concludes the first stage in proving that $\mathcal{H}' \triangleright \mathcal{H}$. The second stage is proven as Lemma A.14, whereas the third stage is proven as Lemma A.18. Therefore, $\mathcal{H}' \triangleright \mathcal{H}$.

We show next that we indeed construct a higher dimensional structure. A careful reading of the enriching construction should answer this question in affirmative. We need to make sure that to each new cell we add all the s and t maps according to its dimension and that we link these maps correctly according to the cubical laws.

The enriching construction proceeds as follows. It takes our initial cell q and its dimension n and the formula that gives the D1 defect. It adds a new cell q_{n+1} of dimension one greater than q and links this with q through the s_{n+1} map. It labels the new cell s.t. the defect of q is repaired. The labeling is not important for our current argument but it is used in the argument for canonicity. To have the new cell q_{n+1} correctly added we need to add n more s maps (i.e., the s_i maps with $1 \leq i \leq n$) and $n+1$ more t maps to it. The s maps are added by the `addSourceMaps` and the t maps are added by the `addTargetMaps`.

Consider now the `addSourceMaps` function which takes as arguments the cell to which it must add the maps and the dimension of this cell, together with two other arguments used for bookkeeping of the cubical laws that need to be added for each cell. More precisely, the m argument records how many cubical laws the q cell enters into. Note that this function (the same as `addTargetMaps`) adds maps only if the dimension of the cell is greater than 0, because, by definition, states in a *HDA* have no maps. `addSourceMaps` adds only $k-1$ maps to its cell argument because one s map has already been added before (e.g., for q_{n+1} we have added the map s_{n+1} and it remains to add the other maps from s_1 to s_n). All these maps link to new cells of dimension one lower (i.e., dimension $k-1$). Actually there are less new cells because some of the s maps must link to already existing cell so to respect the cubical laws. The m argument tells how many s maps should come only from cubical laws and hence, we add only $k-1-m$ new cells. The next loop adds these maps respecting the cubical laws; e.g., for the cell $q_n^{n-1} = s_{n-1}(q)$ we add the map $s_{n-1}(q_n^{n-1})$ as the result of $s_{n-1}(s_n(q))$ (which are cells that have already been added) because of the cubical law $s_{n-1}(s_n(q)) = s_{n-1}(s_{n-1}(q))$. In fact, for the cell q_n^1 each of its s maps links to some existing cell, thus no new cells are added.

Each of the $k-1-m$ new cells are linked with q by the corresponding s_i map. It is also added the s_{k-1} map (i.e., the map with greatest index among the $k-1$ maps that the cell needs). This is done so to respect the cubical

laws $s_i(s_k(q)) = s_{k-1}(s_i(q))$. We now need to recursively add the required s and t maps for the new cell. We call the `addSourceMaps` for this cell q_{k-1}^i of dimension $k-1$ and, depending on the index i in the loop, we specify that $k-1-m-i$ maps should be added directly through the cubical laws and not by using new cells. We must also carry along the node q to which the cubical laws link. We also add the t maps for q_{k-1}^i by calling the `addTargetMaps` function.

The function `addTargetMaps` adds all the t maps of the cell (not one less as the `addSourceMaps` is doing). `addTargetMaps` also tries to respect the cubical laws first, and thus the m argument tells which maps come only from a cubical law like $t_i(t_j(q)) = t_{j-1}(t_i(q))$. For a cell q of dimension k `addTargetMaps` adds $k-m$ new cells of dimension $k-1$ and links each of these cells through a corresponding t_i map. For each new cell a recursive call to `addTargetMaps` is needed to add all the necessary t maps. The s maps of the new cells are added in the end taking care that all the cubical laws of the form $s_i(t_j(q)) = t_{j-1}(s_i(q))$ are respected. All these s maps are linked to cells which come from t maps that have been added by the `addSourceMaps` function before. \square

Lemma A.14. *The new sets of formulas that are added by the enrich algorithm of Lemma A.13 (i.e., at lines 4, 14, 20, 24, 31, 36, 42, 46) are consistent sets.*

Proof. This lemma is essentially the second stage in the proof of the correctness of the enrich construction from Lemma A.13.

Consider only the first set that we construct at line 4. The proof for all the other sets is analogous and simpler. Assume that this set is not consistent, which means two cases: 1) $\psi_1 \wedge \dots \wedge \psi_k \rightarrow \perp$, for $\psi_i \in \tilde{\lambda}(q_{n+1})$ and $\llbracket \rrbracket \psi_i \in \tilde{\lambda}(q)$ with $1 \leq i \leq k$, and 2) $\psi_1 \wedge \dots \wedge \psi_k \rightarrow \neg\phi$, for $\psi_i \in \tilde{\lambda}(q_{n+1})$, $\llbracket \rrbracket \psi_i \in \tilde{\lambda}(q)$, and $\{\}\phi \in \tilde{\lambda}(q)$. For case 1) we know from modal logic that $\llbracket \rrbracket \psi_1 \wedge \dots \wedge \llbracket \rrbracket \psi_k \rightarrow \llbracket \rrbracket (\psi_1 \wedge \dots \wedge \psi_k)$ which, together with the assumption, it means that $\llbracket \rrbracket \psi_1 \wedge \dots \wedge \llbracket \rrbracket \psi_k \rightarrow \llbracket \rrbracket \perp$. This means that $\llbracket \rrbracket \perp \in \tilde{\lambda}(q)$ which is a contradiction with the fact that $\tilde{\lambda}(q)$ contains an existential modality, namely, $\{\}\phi$.⁶ For case 2) we follow a similar argument to obtain $\llbracket \rrbracket \psi_1 \wedge \dots \wedge \llbracket \rrbracket \psi_k \rightarrow \llbracket \rrbracket (\psi_1 \wedge \dots \wedge \psi_k) \rightarrow \llbracket \rrbracket \neg\phi \rightarrow \neg\{\}\phi$. But this is a contradiction because $\tilde{\lambda}(q)$ already contains $\{\}\phi$ and hence would make $\tilde{\lambda}(q)$ inconsistent, contradicting the assumption of potentially canonical labeling of the old HDA .

Note that throughout the rest of the paper when we write $\phi \in \tilde{\lambda}(q)$ we mean that the formula ϕ is part of one of the set constraints in $\tilde{\lambda}(q)$; we use the same notation for the fact that the formula is part of a single constraint when this is clear from the context, as is the case in the paragraph above where we consider only the set constraint build at line 4.

Therefore, we do not need to worry about inconsistencies coming from inside one of the new set constraints that the algorithm builds. It remains to see if any of the formulas in the new set constraints is inconsistent with some formula already existing in the potential label of the cell where the new set constraint is added. This cannot happen at line 4 because the cell q_{n+1} is fresh and has at this point no label attached. Inconsistencies may come from the `addSourceMaps` function that is called recursively in a depth-first manner.

We explain now how this function works and how the source maps are added by the enrich function.

Starting with the defective cell of dimension n the enrich function adds a new cell q_{n+1} of dimension $n+1$ and adds its highest s map, i.e., s_{n+1} . Then it calls the function `addSourceMaps` to add the rest n source maps. This one works in a depth-first manner and adds source maps starting with the highest one. The first call, at line 6 in the body of the enrich itself, adds one less s map, but the other recursive calls add all the maps. Because of the cubical laws, some of the s maps will reach cells that already exist. This is the reason for going in a decreasing order adding first the highest s map, and adding s_1 last. In fact s_1 will have each of its s maps connected to some existing cell. For all the fresh cells that are added, the highest s map will connect to a cell from the old \mathcal{H} . The rest of the cells connect to other fresh cells. This is part of the reason for which `addSourceMaps` works in a depth-first

⁶For the same argument we could have used the existential constraint that is imposed on q at line 5, which implies that the set constraint (i.e., any atom for q containing the universal modalities $\llbracket \rrbracket \psi_i$) must be consistent with $\{\}\top$; which results in a contradiction with the deduced $\llbracket \rrbracket \perp$.

manner when adding the labeling constraints. At the deepest level Q_1 the new cell will connect its only s_1 map to an old cell from \mathcal{H} and its new set constraint will be build from the potential canonical old label. When closing the recursions and going up the levels, the function builds the new set constraints from all these lower cells that are connected through the s maps (one of them is from the old \mathcal{H} , remember). Therefore, the set constraints of all the fresh cells are eventually build only from the labels of old cells.

Consider any of the fresh cells added by the `addSourceMaps` function, i.e., except the q_{n+1} cell added in the body of the `enrich`. Assume that two formulas φ and $\neg\varphi$ come from two different sources containing each a universal formula $\llbracket \rrbracket\varphi$ respectively $\llbracket \rrbracket\neg\varphi$ (as these cannot come from the same source). Because we build these set constraints only from other set constraints from lower level cells reached through s maps it is clear that eventually we reach one of the old cells (from the old \mathcal{H}) which contains both $\llbracket \rrbracket^k\varphi$ and $\llbracket \rrbracket^k\neg\varphi$ (we denoted by $\llbracket \rrbracket^k$ the application of n times of the $\llbracket \rrbracket$ modality) with $k \leq n$. This results in $\llbracket \rrbracket^k \perp$. But because the original q contains $\{\}\varphi$ and each of the old cells reached through an s map has an existential constraint it implies that any of these old cells is consistent with $\{\}\top$, and also the problematic one that by assumption above would have the formula $\llbracket \rrbracket \perp$. Thus we get inconsistency in the old \mathcal{H} , and hence a contradiction. For the first fresh cell q_{n+1} the same argument holds only that we need to treat the case when we actually reach the initial defective formula $\{\}\varphi$. This immediately exhibits the inconsistency with the formula $\llbracket \rrbracket\neg\varphi$, hence the contradiction with the fact that the set constraints of the old \mathcal{H} are consistent.

There is no other way of inconsistencies to creep in the new set constraints for the fresh cells added by the `addSourceMaps` function. We continue the argument for the `addTargetMaps` function.

If in `addSourceMaps` function the accumulation of the set constraints was done in a bottom up fashion after settling the lower cells, i.e., at line 24; now in `addTargetMaps` function the collection is done in a top down fashion, for t maps collecting from all reachable cells that were previously labeled, i.e., at line 31. The `addTargetMaps` function works on adding the new t maps also starting with the highest one and always finishing with t_1 . But many of the maps reach already existing cells: the first **for** loop takes care of such t maps, whereas in the second loop all the s maps reach cells that have been added in the `addSourceMaps` function. For the old cells in the first loop, the function updates the already existing potential labeling by adding new set constraints. For the fresh cells the second loop, at line 36, adds a completely new potential label with one set constraint, and in the next line adds also an existential constraint. All these fresh cells reached through the t maps have their labels updated when the s maps are added. These connect to already existing cells, from where all the boxes have to be accumulated in the label of the new cell, i.e., these are the contents at lines 42 and 46. Note that for some fresh cell the algorithm adds one set constraint coming from each of its source maps.

After this intuitive presentation it is easy to identify where the inconsistencies in the set constraints can come from:

1. either in the first loop at line 31 when collecting a new box constraint having a box formula $\llbracket \rrbracket\varphi$ where the formula $\neg\varphi$ may already be in the potential label $\tilde{\lambda}(t_{k-l}(q))$ as coming from before from some formula $\llbracket \rrbracket\neg\varphi$ in a box constraint of some cell connected to $t_{k-l}(q)$ through a t map;
2. in the second loop when φ comes in the label $\tilde{\lambda}(q_{k-1}^i)$ from the box formula $\llbracket \rrbracket\varphi$ of a cell linked to q_{k-1}^i through a t map and the formula $\neg\varphi$ comes from a box formula $\llbracket \rrbracket\neg\varphi$ at lines 42 or 46 coming from the box constraints of some cell that is connected to q_{k-1}^i through a s map;
3. or when two box formulas $\llbracket \rrbracket\varphi$ and $\llbracket \rrbracket\neg\varphi$ are in the labels of two cells connected to q_{k-1}^i through s maps; i.e., in the third loop at lines 42 or 46.

For 1 it means we are in the first loop, at line 31, in the setting of the cubical law $t_{k+1-m}(t_{k-l+1}(q')) = t_{k-l}(t_{k+1-m}(q'))$, where $q = t_{k+1-m}(q')$ and q' was introduced before by either a previous application of `addTargetMaps` or is one of the cells added by the `addSourceMaps`. Because the two formulas $\llbracket \rrbracket\psi$ and $\llbracket \rrbracket\neg\psi$ may come only from set constraints then the potential label of q' contains both $\llbracket \rrbracket\psi$ and $\llbracket \rrbracket\neg\psi$. We may assume that q' is not added by `addTargetMaps`, but comes from the other two functions; otherwise, we just need to stack several times the $\llbracket \rrbracket$ modality until we reach such a cell, and the reasoning would carry over verbatim. As we argued before, there

exists a cell q'' in the old \mathcal{H} which contains $\llbracket \rrbracket^k \llbracket \rrbracket \psi$ and $\llbracket \rrbracket^k \llbracket \rrbracket \neg \psi$, or in the case when we work with the initial defective formula then $\tilde{\lambda}(q'')$ contains $\llbracket \rrbracket^k \llbracket \rrbracket \psi$ and $\llbracket \rrbracket^{k-1} \{ \} \llbracket \rrbracket \neg \psi$, where $k \geq 2$. Because q'' is from the old \mathcal{H} it means that its labeling is potential canonical and hence has no inconsistencies (i.e., there exist atoms to respect its constraints). But $\llbracket \rrbracket^k \llbracket \rrbracket \psi \wedge \llbracket \rrbracket^k \llbracket \rrbracket \neg \psi \rightarrow \llbracket \rrbracket^k \llbracket \rrbracket \perp$ which contradicts (hence the inconsistency) with the fact that any atom is consistent with $\llbracket \rrbracket^k \langle \rangle \top$. This is because of axiom (A9) applied k times to get $\llbracket \rrbracket^k \langle \rangle \top$ which implies $\llbracket \rrbracket^k \langle \rangle \top$. For the other formulas $\tilde{\lambda}(q'')$ contains $\llbracket \rrbracket^k \llbracket \rrbracket \psi$ and $\llbracket \rrbracket^{k-1} \{ \} \llbracket \rrbracket \neg \psi$ use the same axiom (A9) and infer $\llbracket \rrbracket^{k-1} \{ \} \langle \rangle \perp$ which is inconsistent with the existential constraint in $\tilde{\lambda}(q'')$ that essentially says that the atom should be consistent also with $\{ \} \{ \} \varphi$.

For 2) we are in the second loop and the formula ψ was added at line 36 as coming from $\llbracket \rrbracket \psi \in \tilde{\lambda}(q)$ and the other formulas is added at line 42 (or at line 46 for the same argument) as coming from $\llbracket \rrbracket \neg \psi \in \tilde{\lambda}(t_{i-1}(s_j(q)))$; i.e., we are in the setting of a cubical law $s_j(t_i(q)) = t_{i-1}(s_j(q))$. Assume that q is the defective cell, for otherwise we have less cases to worry about as q would be one of the cells added by `addSourceMaps` and we would have several $\llbracket \rrbracket$ stacked on top of the formulas and the argument would be analog to the one we give below. The fact that $\llbracket \rrbracket \neg \psi \in \tilde{\lambda}(t_{i-1}(s_j(q)))$ means that it comes from a set constraint of $s_j(q)$, i.e., $\llbracket \rrbracket \neg \psi \in \tilde{\lambda}(s_j(q))$. If $s_j(q)$ is not part of the original \mathcal{H} then there is a cell in \mathcal{H} which would have the formula $\llbracket \rrbracket^k \llbracket \rrbracket \neg \psi$, for some $k \geq 1$. We again use the fact that the old \mathcal{H} has a potential canonical labeling and hence is consistent, using the existential constraints, with the formula $\{ \}^k \{ \} \psi$. These two last formulas are inconsistent. Putting them together we obtain $\{ \}^k (\llbracket \rrbracket \neg \psi \wedge \{ \} \psi)$ which by axiom (A7) we get $\{ \}^k (\llbracket \rrbracket \neg \psi \wedge \{ \} \psi) \rightarrow \{ \}^k (\llbracket \rrbracket \perp)$. But this contradicts with the $\llbracket \rrbracket^k \langle \rangle \top$ coming from several applications of axiom (A9).

For 3) the two formulas $\llbracket \rrbracket \psi$ and $\llbracket \rrbracket \neg \psi$ come from two cells introduced by `addSourceMaps` which contain $\llbracket \rrbracket \psi$ respectively $\llbracket \rrbracket \neg \psi$; they cannot come from the old \mathcal{H} . But both these cells reach some cell in the old \mathcal{H} that will contain $\llbracket \rrbracket^k \llbracket \rrbracket \psi$ and $\llbracket \rrbracket^k \llbracket \rrbracket \neg \psi$ and moreover this is consistent with the formula $\{ \}^{k+1} \top$, cf. the existential constraints. The two formulas together imply $\llbracket \rrbracket^k \llbracket \rrbracket \perp$ which together with $\llbracket \rrbracket^k \langle \rangle \top$ implies $\llbracket \rrbracket^k \langle \rangle \perp$, which by using axiom (A7') implies $\llbracket \rrbracket^k \llbracket \rrbracket \perp$ leading to an inconsistency with $\{ \}^{k+1} \top$.

Both functions always take care to add the existential constraints for any map that is added. \square

Definition A.15 (descents). Define the relation $\xrightarrow{s} \subseteq Q \times Q$ as $q \xrightarrow{s} q'$ iff $\exists s_i : s_i(q) = q'$. Define $\xleftarrow{t} \subseteq Q \times Q$ as $q \xleftarrow{t} q'$ iff $\exists t_i : t_i(q') = q$. Define $\xleftrightarrow{st} = \xrightarrow{s} \cup \xleftarrow{t}$ (and call its elements descent steps), and \xleftrightarrow{st}^* as their reflexive transitive closure. We call a sequence (i.e., composition of relations) from \xleftrightarrow{st}^* a descent chain. A descent chain is maximal if no more descent steps can be added.

Descent chains are somehow the opposite of *paths* in HDAs, cf. Definition 3.4.

Lemma A.16. For the enrich algorithm for any of the new cells that are added, for any of its immediate starting descends it will eventually end up descending in one of the old cells of \mathcal{H} . Formally: $\forall q' \in \mathcal{H}' \setminus \mathcal{H}, \forall q' \xleftrightarrow{st}$, $\exists q \in \mathcal{H} : q' \xleftrightarrow{st} \circ \xleftrightarrow{st}^* q$.

Proof. The first fresh cell is added at line 2 and is directly linked through s_{n+1} , at line 3, to the original defective cell from the old \mathcal{H} .

It remains to check that all other s and t maps of this cell are eventually reaching the old model \mathcal{H} . We do this inductively by going down the recursion calls until we find the minimal single descent steps. In particular we have to check only the s maps because from this initial fresh cell q_{n+1} only \xrightarrow{s} steps are possible.

If $s_{n+1}(q_{n+1})$ is linked to the cell q in the old \mathcal{H} , then for all the other $s_i(q_{n+1})$ their s_n map is linked to the s_i map of q , hence reaching in one \xrightarrow{s} step \mathcal{H} . We used the cubical law $s_n(s_{n+1}(q_{n+1})) = s_n(s_n(q_{n+1}))$.

We use the similar law $s_{n-1}(s_n(s_{n+1}(q_{n+1}))) = s_{n-1}(s_{n-1}(s_n(q_{n+1})))$ to argue that taking the descent step using the source s_{n-1} reaches the old \mathcal{H} in two steps. The same reasoning is carried over inductively until the last recursion call.

In conclusion, all fresh cells added by the `addSourceMaps` reach the old \mathcal{H} through any of their sources by following a descent chain formed only of descent steps from \xrightarrow{s} , which, depending on the index of the source, take longer or shorter to reach \mathcal{H} .

The other fresh cells are added in the `addTargetMaps`, at line 28. It is easy to see that all their immediate \xleftarrow{t} possible steps can eventually reach \mathcal{H} . An easy inductive reasoning suffices for this argument. Start with the cells added in the `addSourceMaps` function or at line 7 in the body of the `enrich` function itself. All these reach in one \xleftarrow{t} step a cell that we argued before that it can reach \mathcal{H} through a chain of only \xrightarrow{s} . For the other cells added at deeper recursion calls inside `addTargetMaps` we can reach the cells from before, which have a descent chain to \mathcal{H} .

It remains to show that for all the fresh cells added in the `addTargetMaps` their s maps also lead to \mathcal{H} , i.e., that starting also with a \xrightarrow{s} step also leads eventually to a descent chain to \mathcal{H} . This is done also inductively starting with the cells that are added in the first call to `addTargetMaps` function, and not inside its body (i.e., this step also considers the first calls inside the `addSourceMaps` function). Therefore, we consider some cell q' which we proved that it eventually reaches \mathcal{H} ; this cell has a target, say t_k , to the fresh cell q that we are concerned with and itself has a source to some other cell $s_j(q) = q''$. Depending on k, j we use the following cubical laws: if $j < k$ then $s_j(t_k(q')) = t_{k-1}(s_j(q'))$; if $k \leq j$ then $t_k(s_{j+1}(q')) = s_j(t_k(q'))$. Thus, in any case we can have a \xleftarrow{t} step from q'' to s_j (or s_{j+1} depending on the case), but these cells can reach \mathcal{H} , because they are reached from the initial q' through a \xrightarrow{s} step. This base case is finished.

For cells added at deeper recursion calls, inside `addTargetMaps`, we use the same cubical laws and reach cells that we just proven in the step before that can reach \mathcal{H} . Depending on the indexes of the s maps, the descending chains are longer or shorter. \square

Note that in the proof of Lemma A.14 we made heavy use of the fact that we could go down a descent chain that was made of only \xrightarrow{s} steps. Because of this we were stacking up $\llbracket \cdot \rrbracket$ modalities. We will shortly make precise this method of stacking modalities depending on the descent chain and we will see more use of it and in more varied settings.

Corollary A.17.

1. For any fresh cell added by the `enrich` algorithm there exists a maximal descending chain and this one reaches a cell in the old \mathcal{H} that can make no \xrightarrow{s} and no \xleftarrow{t} steps.
2. For any fresh cell, any descent chain that reaches \mathcal{H} can be completed to the maximal descent chain.
3. For any fresh cell that has one descent chain starting with \xleftarrow{t} and one starting with \xrightarrow{s} , both these descent chains eventually reach the same cell in the old \mathcal{H} and use the same number of descent steps and the same number of \xrightarrow{s} steps (hence the same number of \xleftarrow{t} steps also).
4. For two cells connected as $s_i(q) = q'$ then the maximal descending chain of q is one greater than the maximal descending chain of q' .

Proof. \square

Lemma A.18. For the `enrich` algorithm of Lemma A.13 all the new existential constraints that are added to the fresh cells or to cells from the old HDA are consistent with the set constraints of that cell.

Proof. This lemma essentially makes the third stage of the proof that the HDA built by the `enrich` construction extends the old HDA. This proof is based on the fact that the set constraints of each cell are consistent, cf. Lemma A.14. We will make use of the fact that we enrich an old \mathcal{H} which has a potential canonical labeling, as we did in the proof of Lemma A.14. The proof is by *reductio ad absurdum* and assumes for an existential constraint $\{\} \lambda(q)$ in the $\tilde{\lambda}(q')$ there exists a formula $\llbracket \cdot \rrbracket \neg \psi$ in the set constraints of $\tilde{\lambda}(q')$ for which the formula ψ is in some set constraint of $\tilde{\lambda}(q)$.

These two formulas come from other set constraints being under some $\llbracket \rrbracket$ or $\llbracket \rrbracket$ box modality; and these bigger formulas in turn come from other set constraints added by the algorithm. And so on until we reach the old \mathcal{H} . From here we only know that the labeling is potential canonical; and we will use this in the proof. Therefore there are many ways that the assumed formulas may have come from, and we need to find a way to treat all these different ways.

First we checked all the cases by hand for the particular application of enrich when the old cell is of dimension 2 and hence the new cell that needs to be added (with all its maps) is of dimension 3. Finding a clear pattern in these cases with varied length of the constraint propagation was tedious.⁷ The definitions and results above about descent chains are the basis of the general proof pattern that we will develop now further. These chains relate to the histories of a cell and to the notions of adjacency and homotopy.

We take any s map introduced by the algorithm, i.e., $s_i(q) = q'$, and make the assumption from above. We will arrive at an inconsistency in the old \mathcal{H} , hence the contradiction. (The same proof method works for the t maps and an analog assumption as above only that we use $\llbracket \rrbracket$ instead of $\llbracket \rrbracket$.) Note that from cell q we can make a \xrightarrow{s} descent step to reach q' . From Corollary A.17 we know that from the cell q' , hence also from q , there exists some descent chain reaching \mathcal{H} . If the descent chain is empty, i.e., $q' \in \mathcal{H}$, and if all the descent chains of q consist in the single descent step to q' then the result is trivial. This case corresponds to when the algorithm is at the most deep recursion call. If there are other descent steps starting from q then we are in a nontrivial case.

From Corollary A.17 we know that any two different descent chains starting from q will eventually end up in the same cell. Moreover, we know that any such descent chain eventually reaches the \mathcal{H} or the newly added cell q_{n+1} at line 2 in the algorithm. We can argue that is enough to consider reaching this cell instead of reaching the original \mathcal{H} .

The proof method takes two such descent chains starting from q : one going first through q' and the other going through some other different cell. For each of these chains we stop at the first cell from \mathcal{H} or q_{n+1} . The idea is that until there we are walking through the fresh cells added by the algorithm, and hence we collect boxes on the way.

Example A.1. Take the example of q which has one descending chain $q \xleftarrow{t} q''$. This is to say that the formula ψ comes from a formula $\llbracket \rrbracket \psi$ in a set constraint of q'' . For a descent chain $q \xrightarrow{s} \xleftarrow{t} q''$, where the first step does not go through q' , it means that ψ comes from $\llbracket \rrbracket \llbracket \rrbracket \psi \in \tilde{\lambda}(q'')$. The fact that the other descent chain that we consider from q goes through q' with a \xrightarrow{s} step should mean that $\llbracket \rrbracket \psi$ is in the set constraints of q' but this contradicts the assumption that $\llbracket \rrbracket \neg \psi$ is there and that there actually exists an s map out of q ; this contradiction will come syntactically as an inconsistency in the potential canonical labeling of the old \mathcal{H} .

To such a descent chain that starts from q we associate a formula as follows: considering $\psi \in \tilde{\lambda}(q)$, for $q \xrightarrow{s} q'' \xleftarrow{st}^* \xrightarrow{st}^*$ then $\llbracket \rrbracket \psi \in \tilde{\lambda}(q'')$. We continue until the end of the chain where in the case of a \xleftarrow{t} descent step eg. $q'' \xleftarrow{t} q''' \xleftarrow{st}^* \xrightarrow{st}^*$ we have $\llbracket \rrbracket \llbracket \rrbracket \psi \in \tilde{\lambda}(q''')$. Both these chains reach eventually the same cell in \mathcal{H} , cf. Corollary A.17. Moreover, the chain that reaches \mathcal{H} faster can continue through the inside of \mathcal{H} until reaching the end cell of the other chain. This traversing of the old \mathcal{H} is done under the existential constraints in the potential canonical labeling of \mathcal{H} , therefore we can extend the corresponding formula with existential modalities. In the common end cell we have now two formulas, one made only of box modalities and the other which may also have a stack of existential modalities, and both have to be consistent, as being part of the old \mathcal{H} . We actually show that these two are inconsistent or cannot be grown to an atom, i.e., lead to an inconsistency in the axiomatic system of *HDML*.

Thus we work with two chains starting from q and ending in some common cell in \mathcal{H} , and to each chain we associate a formula: one adds modalities to ψ (as being in $\tilde{\lambda}(q)$) and the other adds modalities to $\neg \psi$ (as coming from the chain that goes through q' which we assumed to have a formula $\llbracket \rrbracket \neg \psi$). We denote descent steps

⁷Even if for the cells or maps added/reached at the most inner recursion depth in the enrich algorithm the cases were easy to check or trivial, it is not possible to use an inductive reasoning in this way because at outer recursion levels the mesh of maps that connect to some particular cell becomes too complex.

that are inside the \mathcal{H} , and which are associated with existential modalities, by \xrightarrow{s} respectively \xleftarrow{t} . There is an equal number of \xrightarrow{s} in each chain (either universal or existential) and hence an equal number of \xleftarrow{t} steps also, cf. Corollary A.17. This translates into the formulas also. The purpose is to change these chains so that the descent steps match one by one. If one existential matches one universal then we obtain an existential step that leads to the inconsistency more easy. Also, the purpose is to move as much as possible of the \xrightarrow{s} steps to the end of the chain and the \xleftarrow{t} steps to the beginning of the chain. In the end we will arrive at a contradiction with the fact that in the initial cell which the algorithm starts with there is the formula $\{\}\varphi$ and hence, by the existential constraints in \mathcal{H} , all lower cells are consistent with $\{\}^k\varphi$ depending on the distance from the initial cell.

For example:

$$\begin{array}{ll} \psi \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} & \text{associate } \{\}\{\}\{\}\psi \\ \neg\psi \xrightarrow{s} \xleftarrow{t} \xrightarrow{s} & \text{associate } \{\}\{\}\{\}\neg\psi \end{array}$$

Apply axiom (A10) to get $\{\}\{\}\{\}\psi \wedge \{\}\{\}\{\}\neg\psi$ which by modal reasoning becomes $\{\}\{\}\{\} \perp$, but by modal reasoning and axiom (A9) we have as validity $\{\}\{\}\{\}^k \top$ where in our case we use it for $k = 2$ to get $\{\}\{\}\{\} \perp$. This results in a contradiction with the fact that we can always start at least to reach a cell with φ .

For all the patterns that we find in the descent chains there is some axiom associated which helps transform the formulas into the needed ones; we will say that they transform the chains into the proper form. Below we give the patterns with the associated formulas and axioms:

$$\begin{array}{llll} \xrightarrow{s} \xleftarrow{t} \xrightarrow{s} & \text{is } \{\}\{\}\{\} \xrightarrow{(A10)} \{\}\{\}\{\} & \text{is } \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} \\ \xleftarrow{t} \xrightarrow{s} \xleftarrow{t} & \text{is } \{\}\{\}\{\} \xrightarrow{(A10')} \{\}\{\}\{\} & \text{is } \xleftarrow{t} \xleftarrow{t} \xrightarrow{s} \\ \xrightarrow{s} \xleftarrow{t} \xrightarrow{s} & \text{is } \langle \rangle \{\}\{\} \xrightarrow{(A7')} \{\}\{\} & \text{is } \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} \\ \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} & \text{is } \{\}\{\} \xrightarrow{(A7)} \{\}\{\} & \text{is } \xrightarrow{s} \xleftarrow{t} \xrightarrow{s} \end{array}$$

Note that the last pattern does not bring a \xleftarrow{t} more close to the beginning of the chain, but does the opposite. This is the case when the other three patterns do not occur but we can match the \xrightarrow{s} that is brought closer to the beginning to a \xrightarrow{s} , therefore combining the two (by modal reasoning) into a \xrightarrow{s} applied to \perp which just makes the descent step disappear into \perp ; i.e., $\langle \rangle \perp \leftrightarrow \perp$.

There may be patterns that are not matched by any of the above, like eg.:

$$\begin{array}{ll} \psi \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} \xleftarrow{t} & \text{associate } \langle \rangle \{\}\{\}\{\}\psi \\ \neg\psi \xrightarrow{s} \xleftarrow{t} \xleftarrow{t} \xrightarrow{s} & \text{associate } \{\}\{\}\{\}\neg\psi \end{array}$$

The pattern in the lower chain is not matched by any of the four patterns above because there are more than one $\{\}$ stacked on top of each other, i.e., are two consecutive \xleftarrow{t} steps surrounded by \xrightarrow{s} steps. Nevertheless, such patterns can be *broken* s.t. the new chains can be matched by the four main patterns above. Breaking such patterns (also with more than two consecutive \xleftarrow{t}) is done with the use of axioms (A9), (A8'), (A8), possibly applied several times. In the particular case above, because we have the first formula then axiom (A9) is applied (for $n = 1$) to get $\{\}\{\}\langle \rangle \top$ to which we can apply axiom (A8') to get $\{\}\{\}\langle \rangle \top$. This formula now breaks the second chain in the sense that one \xleftarrow{t} is transformed into an existential one \xleftarrow{t} ; i.e., $\{\}\{\}\{\}\neg\psi \wedge \{\}\{\}\langle \rangle \top \rightarrow \{\}\{\}\langle \rangle \{\}\neg\psi$. To this chain now we can apply the third and then the first pattern from above to obtain $\{\}\{\}\{\}\langle \rangle \neg\psi$, i.e., the chain $\neg\psi \xleftarrow{t} \xleftarrow{t} \xrightarrow{s} \xrightarrow{s}$. To the first chain we could apply the third pattern two times to get $\{\}\{\}\{\}\langle \rangle \psi$, i.e., the chain $\psi \xleftarrow{t} \xleftarrow{t} \xrightarrow{s} \xrightarrow{s}$. It is clear that the two formulas contradict with the fact that the current cell of \mathcal{H} must be consistent, by the existential constraints, with $\{\}\{\}\{\}\varphi$ because $\{\}\{\}\{\}\langle \rangle \neg\psi \wedge \{\}\{\}\{\}\langle \rangle \psi \rightarrow \{\}\{\}\{\}\langle \rangle \perp \rightarrow \{\}\{\}\{\} \perp$. In terms of descent

chains the two chains match step by step as having the same s or t label and we match either two universal steps, like \xrightarrow{s} , or one universal step with one existential step which yield an existential step, like \xleftarrow{t} with $\xleftarrow{t} \dashrightarrow$. Such match of descent chains yields the inconsistency with the chain $\top \dashrightarrow \dashrightarrow \dashrightarrow \dashrightarrow$.

For another example of unmatched patterns in chains consider:

$$\begin{array}{ll} \psi \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} \xleftarrow{t} \dashrightarrow \xrightarrow{s} & \text{associate } \{\} \langle \rangle [\![\![\![\![\![\psi \\ \neg \psi \xrightarrow{s} \xleftarrow{t} \xleftarrow{t} \xrightarrow{s} \xrightarrow{s} & \text{associate } [\![\![\![\![\![\neg \psi \end{array}$$

This is the same as the example before only that each chain is extended with one existential \dashrightarrow step respectively an universal \xrightarrow{s} . The same way of breaking the pattern using axioms (A9) and (A8') is used here also only that because we do not have the formula $\langle \rangle \top$ we use twice (A9) to get $[\![\![\![\![\![\langle \rangle \top$ which by axiom (A8') we obtain the breaking pattern $[\![\![\![\![\![\langle \rangle \top$ with the associated chain $\xleftarrow{t} \dashrightarrow \xleftarrow{t} \xrightarrow{s} \xrightarrow{s}$. This breaks the second chain into $\xrightarrow{s} \xleftarrow{t} \dashrightarrow \xleftarrow{t} \xrightarrow{s} \xrightarrow{s}$ (or the formula becomes $[\![\![\![\![\![\langle \rangle \neg \psi$) to which we can apply the pattern 3 and then 1 to obtain the chain $\xleftarrow{t} \dashrightarrow \xleftarrow{t} \xrightarrow{s} \xrightarrow{s}$. To the first one applies pattern 3 two times to obtain $\xleftarrow{t} \dashrightarrow \xleftarrow{t} \dashrightarrow \xrightarrow{s} \xrightarrow{s}$ so the two chains match step by step. Whenever we are in a situation like this when the two modified chains end up in an existential \dashrightarrow step and a corresponding universal one, we can just remove these two steps because it basically says that there exists this reachable cell where both formulas $[\![\![\![\![\![\langle \rangle \neg \psi$ and $[\![\![\![\![\![\langle \rangle \psi$ hold. These result in an inconsistency with the existential constraints again. \square

For a D2 defect, the *lifting construction* lifts the defective cell and all the cells that are connected to it by some s or t map, one level up by adding one new s and t map to each of them. The label of the new t map will be the one repairing the D2 defect. The cubical laws make sure that these new maps reach only new cells; none of the old cells (that are lifted) are involved in these new instances of the cubical laws. We need to be careful how we label all these new cells s.t. the canonicity is respected for the new lifted *HDA*.

The lifting construction is more involved than the enriching construction. We still label the cells with atoms in the end but during the construction the constraints that the atom has to satisfy are changed. This is why we keep the set of all atoms that satisfy the constraints as possible candidates for the final labeling. This means that we are still working with atoms (i.e., maximal consistent sets of formula) but we do not settle on one particular atom until we have finished the construction.

Lemma A.19 (lifting construction). *For a canonical model \mathcal{H} , there exists a construction (see Appendix), which we call lifting of the \mathcal{H} wrt. q and a formula $\langle \rangle \phi \in \lambda(q)$, builds a model \mathcal{H}' which is canonical and extends \mathcal{H} (i.e., $\mathcal{H}' \triangleright \mathcal{H}$).*

Proof. The lifting construction is the following:

```

1 function lift(n,q, $\varphi$ ){
2   addTargetMap(n,q,{ $\varphi$ }, $\emptyset$ ); // add the source map
3   addSourceMap(n,q); // add the target map
4   for(all cells  $q'$  with  $q' \in Q_m$ ){
5     lift(m, $q'$ , $\emptyset$ ) // lift all other cells
6   }
7   function addTargetMap(k,q,S1,S2){
8      $Q_k := Q_k \cup \{q_k\}$ ; // fresh cell
9     update map  $t_{k+1}$  s.t.  $t_{k+1}(q) = q_k$ ;
10    add constraints  $\tilde{\lambda}(q_k) = S1 \cup \{\phi \mid []\phi \in \tilde{\lambda}(q)\} \cup \{\phi \mid []\phi \in S2\}$ ;
11    add constraints  $\tilde{\lambda}(q) = \tilde{\lambda}(q) \cup \langle \rangle \tilde{\lambda}(q_k)$ ;
12    for(i=1 to k){
13       $r_{k-1}^i := \text{addTargetMap}(k-1, s_i(q), \emptyset, \emptyset)$ ;
14      update map  $s_i$  s.t.  $s_i(t_{k+1}(q)) = r_{k-1}^i$ ;
15      add constraints  $\tilde{\lambda}(q_k) = \tilde{\lambda}(q_k) \cup \{\phi \mid []\phi \in \tilde{\lambda}(r_{k-1}^i)\}$ ;
16      add constraints  $\tilde{\lambda}(r_{k-1}^i) = \tilde{\lambda}(r_{k-1}^i) \cup \{\}\widehat{\lambda}(q_k)$ ;
17    }
18    for(i=1 to k){
19       $q_{k-1}^i := \text{addTargetMap}(k-1, t_i(q), \emptyset, \lambda(q_k))$ ;
20      update map  $t_i$  s.t.  $t_i(t_{k+1}(q)) = q_{k-1}^i$ ;
21      add constraints  $\tilde{\lambda}(t_{k+1}(q)) = \tilde{\lambda}(t_{k+1}(q)) \cup \langle \rangle \widehat{\lambda}(q_{k-1}^i)$ ;
22    }
23    return  $q_{k-1}$ ;
24  }
25  function addSourceMap(k,q){
26     $Q_k := Q_k \cup \{q_k\}$ ; // fresh cell
27    update map  $s_{k+1}$  s.t.  $s_{k+1}(q) = q_k$ ;
28    add constraints  $\tilde{\lambda}(q_k) = \{\}\widehat{\lambda}(q)$ ;
29    for(i=1 to k){ // add a fresh  $s_k$  map to each  $t_i(q)$ 
30       $r_{k-1}^i := \text{addSourceMap}(k-1, t_i(q))$ ;
31      update map  $t_i$  s.t.  $t_i(s_{k+1}(q)) = r_{k-1}^i$ ;
32      add constraints  $\tilde{\lambda}(q_k) = \tilde{\lambda}(q_k) \cup \langle \rangle \widehat{\lambda}(r_{k-1}^i)$ ;
33    }
34    for(i=1 to k){ // add a fresh  $s_k$  map to each  $s_i(q)$ 
35       $q_{k-1}^i := \text{addSourceMap}(k-1, s_i(q))$ ;
36      update map  $s_i$  s.t.  $s_i(s_{k+1}(q)) = q_{k-1}^i$ ;
37      add constraints  $\tilde{\lambda}(q_{k-1}^i) = \tilde{\lambda}(q_{k-1}^i) \cup \{\}\widehat{\lambda}(q_k)$ ;
38    }
39     $Q_k \setminus \{q\}; Q_{k+1} \cup \{q\}$ ; // move the cell one level up
40    return  $q_{k-1}$ ;
41  }

```

The proof has several stages:

1. We need to show that the enriched model is an extension of the old model (i.e., $\mathcal{H}' \triangleright \mathcal{H}$), which amounts to:
 - (a) first showing that the structure of the old *HDA* is untouched, i.e., all old cells and maps are in place;
 - (b) then showing that all set constraints are still consistent sets;
 - (c) and third showing that all the new existential constraints do not contradict with the box constraints.

Basically the steps (1b) and (1c) are corresponding to Lemma A.11 to show that the new potential labeling is still potential canonical, i.e., that there still exists a way of instantiating the constraints to atoms.
2. The next stage shows that \mathcal{H}' is a model indeed, i.e., that all the maps are in place and all necessary cubical laws are respected.

3. The last stage shows that the enriched model does not have the old defect and that no new defects are introduced in the potential labeling of the initial cell q .

First remark that we do not change the initial shape of the original \mathcal{H} ; we only add fresh cells and fresh maps for these cells; we also add maps to old cells connected to new cells. This concludes the first stage in proving that $\mathcal{H}' \triangleright \mathcal{H}$. The second stage is proven as Lemma A.14, whereas the third stage is proven as Lemma A.18. Therefore, $\mathcal{H}' \triangleright \mathcal{H}$.

We show next that we indeed construct a higher dimensional structure. A careful reading of the enriching construction should answer this question in affirmative. We need to make sure that to each new cell we add all the s and t maps according to its dimension and that we link these maps correctly according to the cubical laws.

Note that the algorithm finishes with a completely new layer of cells denoted Q_{-1} ; in the end of the construction we have to rename all the layers Q_i into Q_{i+1} to make justice to the cells that reside there which have now dimension $i + 1$ as we added one s and one t map to each.

Note that the construction terminates iff q is in a hypercube of finite dimension and in this case we ignore all the cells outside this cube. (The construction always terminates when we use it in the repair lemma A.23.)

Clearly the two functions do not change the labels nor the shape of the old \mathcal{H} and hence the lifted \mathcal{H}' has all the structure of \mathcal{H} .

Now we show that the lifting constructs indeed a *HDA*. This means that we must make sure that all the (new) cells have the right number of s and t maps and that all the cubical laws are respected.

The `lift` function takes as input the reference cell q and its dimension n together with the formula φ that causes the defect (i.e., $\langle \varphi \in \tilde{\lambda}(q) \rangle$). Then the function adds one t map and one s map to q by calling `addTargetMap` and `addSourceMap` respectively. These two functions add one new cell and link it with either a t or an s map. All other cells that are connected to q must also be lifted, which is done in the loop of the `lift` function.

Consider now the `addTargetMap` which takes as arguments the cell q (and its dimension k) to which the new t map needs to be added. It also takes two sets of formulas which are used to construct the label of the new cell and of the other new cells connected to it recursively. We do not discuss here the labeling because we do this in the Lemmas ?? and ?. The rest of the proof is concerned with the geometric structure of the extended \mathcal{H}' .

The `addTargetMap` function adds the new t_{k+1} map to q , which is the map with the largest index (i.e., the new index showing that the q cell has now dimension one greater, $k + 1$). It links this with a new cell q_k of dimension one lower than the new dimension of the input cell q . The first loop does two operations. First it lifts all the old cells linked to q by an s map (i.e., $s_i(q)$) by adding one t map to each; i.e., it invokes `addTargetMap` recursively. Then, all these cells enter under new cubical laws that involve the s maps of the newly added q_k cell. In this way we also add all the necessary s maps of q_k and also respect the new cubical laws $s_i(t_{k+1}(q)) = t_k(s_i(q))$.

In the second loop of `addTargetMap` we add the new t_k map to each old cell linked to q by a t_i map; i.e., in the recursive invocation of `addTargetMap`. At the same time we add all the t maps for the new q_k cell and link these through the cubical laws $t_i(t_{k+1}(q)) = t_k(t_i(q))$.

The construction goes recursively at lower levels until reaching cells of dimension 0. These are the last cells lifted to have dimension 1. Here the recursion stops.

Consider now the similar function `addSourceMap` which adds one s map to the input cell q of dimension k to make it now of dimension $k + 1$. Therefore, it adds the map $s_{k+1}(q) = q_{k-1}$. This is also the place where the lifted cells are actually moved to the rightful layer Q_{k+1} , at the end of the function (i.e., line 39), after both target and source maps have been added.

In the first loop `addSourceMap` adds a new s_k maps to all the old cells linked to q by a t map. This finishes what we started in the second loop of `addTargetMap`, i.e., finishes lifting all the $t_i(q)$ cells. It also takes care to respect all the new cubical laws $t_i(s_{k+1}(q)) = s_k(t_i(q))$ and, hence, to add the t_i maps to q_k .

The second loop complements what we started in the first loop of `addTargetMap`. We finish adding the s_k maps to all the $s_i(q)$ cells. It also adds all the s maps to q_k and respects the new cubical laws $s_i(s_{k+1}(q)) = s_k(s_i(q))$.

In conclusion, all the cells of the old \mathcal{H} have been added one new t and s map, each reaching a new cell. To all these new cells all the t and s maps have been added and linked according to the new cubical laws.

□

Lemma A.20. *The new sets of formulas that are added by the lift algorithm of Lemma A.19 (i.e., at lines 10 and 15) are consistent sets.*

Proof. The only place where box constraints are added by the lift function is in addTargetMap: first at line 10 and then repeatedly in the loop at line 15.

The set S1 is not empty only when the function is applied to the initial cell q from the statement of the lemma. The lemma assumes that $q \in Q_n$ is of dimension n , denote it q_n for this part of the proof, and it contains $\langle \rangle \varphi \in \tilde{\lambda}(q_n)$ for which all of its n existing s maps contain $\neg\varphi$. This means that if before $\tilde{\lambda}(q_n)$ was consistent with $\langle \rangle n$ now we need to write $\langle \rangle n+1$. Because of axiom (A5) and Lemma A.2(ii) it means that $\tilde{\lambda}(q_n)$ is consistent also with $\langle \rangle^{n+1} \top$. (As a side remark, we use Lemma A.2(ii) tacitly in many places during the proofs of the two constructions lemmas.)

The first call to addTargetMap($n, q_n, \{\varphi\}, \emptyset$) makes use only of S1 and constructs the set $\{\varphi\} \cup \{\psi \mid []\psi \in \tilde{\lambda}(q_n)\}$. This set is associated to $q_{n-1} = t_{n+1}(q_n)$. The proof is easy for this case and uses arguments as in the proof before: if we assume $\psi_1 \wedge \dots \wedge \psi_k \rightarrow \perp$ then we get that $[]\perp \in \lambda(q_n)$ which is a contradiction as $\lambda(q_n)$ is an atom containing $\langle \rangle \varphi$; if we assume $\psi_1 \wedge \dots \wedge \psi_k \rightarrow \neg\varphi$ then we get that $[]\neg\varphi \in \lambda(q_n)$ which is again a contradiction.

The second call to addTargetMap is made for each s map of a cell q (in the first loop of the body of the addTargetMap) and it uses only the set S2. This means that it labels a cell $q_{n-1} = t_{n+1}(q)$ with a set $\{\psi \mid []\psi \in \lambda(q)\}$. Assume $\psi_1 \wedge \dots \wedge \psi_k \rightarrow \perp$ which means that $[]\perp \in \lambda(q)$. This is a contradiction because $\lambda(q)$ is an atom and it contains at least one diamond formula. This is because q has dimension at least 1 (as it has at least one t map) and we show that any cell of dimension n , with $n \geq 1$, has a formula $\langle \rangle^n \top \in \lambda(q)$. We showed before that the topmost cell q_n has the formula $\langle \rangle^{n+1} \top$ in its label and hence it is of dimension $n+1$. This means that any cell reached through one of its t maps will have the formula $\langle \rangle^n \top$ because of axiom (A8') which says that $\langle \rangle \langle \rangle^n \top \rightarrow []\langle \rangle^n \top$ it means that $[]\langle \rangle^n \top \in \lambda(q_n)$ and by the construction of their labels it means that $\langle \rangle^n \top \in \lambda(t_j(q_n))$. This holds for any cell reached through any number of applications of t maps. On the other hand, the cells reached through an s map from q_n , by canonicity, they contain $\{\langle \rangle^{n+1} \top$, which, by axiom (A9') it means that $\langle \rangle^n \top \in \lambda(s_j(q_n))$.

It remains to see that with each iteration of the first loop the updated label remains a consistent set. This update is necessary when we are trying to respect the cubical laws of the form $s_i(t_{k+1}(q)) = t_k(s_i(q))$. The proof of this part follows an inductive argument, where the basis was just proven above and the inductive case is for some i iteration, where we consider that the label is a consistent set (and all the other labels that the construction uses have been built already and, hence, are atoms). Assume that for some $[\Box]\psi \in \lambda(t_k(s_i(q)))$ there has already been added the $\neg\psi$ to $\lambda(t_{k+1}(q))$. This has happened in two cases: first if $\neg\psi$ comes from $\lambda(q)$, i.e., $[]\neg\psi \in \lambda(q)$ which by canonicity it means that $\{\Box[]\neg\psi \in \lambda(s_i(q))$. On the other hand we also have that $\langle \rangle[\Box]\psi \in \lambda(s_i(q)) \xrightarrow{(A7')} [\Box]\langle \rangle\psi \in \lambda(s_i(q))$. Together with the above it means that $\{\Box([]\neg\psi \wedge \langle \rangle\psi) \rightarrow \{\Box\langle \rangle(\neg\psi \wedge \psi) \xrightarrow{(A2), (A2')} \perp \in \lambda(s_i(q))$ which is a contradiction with the fact that $\lambda(s_i(q))$ is an atom. The second case is when $\neg\psi$ has been added in a previous iteration, i.e., $[\Box]\neg\psi \in \lambda(s_j(t_{k+1}(q)))$ with $1 \leq j < i$. But this means that each of these two cells must have at least one s map and enter the cubical law $s_j(s_i(t_{k+1}(q))) = s_{i-1}(s_j(t_{k+1}(q))) = q''$. By the canonicity of these lower cells we have that $\{\Box[\Box]\neg\psi \in \lambda(q'')$ and $\{\Box[\Box]\psi \in \lambda(q'')$. From axiom ?? we have that $[\Box]\{\Box\psi \in \lambda(q'')$ and thus $[\Box]\{\Box\psi \wedge \{\Box[\Box]\neg\psi \rightarrow \{\Box\}(\{\Box\psi \wedge [\Box]\neg\psi) \rightarrow \{\Box\}(\psi \wedge \neg\psi) \xrightarrow{(A2)} \perp \in \lambda(q'')$ which is a contradiction.

The application of addTargetMap in the second loop uses the S3 set also and we are looking at cubical laws of type $t_i(t_{k+1}(q)) = t_n(t_i(q))$ where $S2 = \lambda(t_i(q))$ and $S3 = \lambda(t_{k+1}(q))$. Assume, for the sake of contradiction, that we have $[]\psi \in \lambda(t_{k+1}(q))$ and $[]\neg\psi \in \lambda(t_i(q))$. By canonicity it means that $\langle \rangle[]\psi \in \lambda(q)$ and $\langle \rangle[]\neg\psi \in \lambda(q)$ and from axiom (A6) we have $[]\langle \rangle\neg\psi \in \lambda(q)$. This means that $[]\langle \rangle\neg\psi \wedge \langle \rangle[]\psi \rightarrow \langle \rangle(\langle \rangle\neg\psi \wedge []\psi) \rightarrow \langle \rangle\langle \rangle(\psi \wedge \neg\psi) \xrightarrow{(A2')} \perp \in \lambda(q)$ which is a contradiction. □

Lemma A.21. *For the enrich algorithm of Lemma A.19 all the new existential constraints that are added to the fresh cells or to cells from the old HDA are consistent with the set constraints of that cell.*

Proof. Assume that for the lifted HDA the *second canonicity condition is broken*; i.e., consider $q \in Q_n$ and assume $t_i(q) = q'$ for which $\varphi \in \lambda(q')$ and $\langle \varphi \rangle \notin \lambda(q)$, which is the same as $\neg \langle \varphi \rangle \in \lambda(q)$. We take cases after q .

First, clearly, if $q, q' \in \mathcal{H}$ (meaning that $1 \leq i \leq n-1$) then the canonicity is assured by the statement of the lemma (i.e., \mathcal{H} is canonical).

Second, $q \in \mathcal{H}$ and q' is added by `addTargetMap` as the new cell linked to q by $t_n(q) = q'$. Now we take sub-cases depending on where does the φ formula come from.

- If $\varphi \in S1$; this is the case when q is the initial cell from the statement of the lemma and hence it cannot be that $\neg \langle \varphi \rangle \in \lambda(q)$.
- If $\varphi \in \{\varphi \mid \llbracket \varphi \rrbracket \in S2\}$ then $\llbracket \varphi \rrbracket \in \lambda(q)$ and the assumption says that $\llbracket \neg \varphi \rrbracket \in \lambda(q)$. This is a contradiction as $\llbracket \varphi \rrbracket \wedge \llbracket \neg \varphi \rrbracket \rightarrow \llbracket (\varphi \wedge \neg \varphi) \rrbracket \rightarrow \llbracket \perp \rrbracket \in \lambda(q)$ which is not possible because, as we showed before, $\lambda(q)$ contains at least one existential formula, i.e., $\langle \varphi \rangle^k \top$, where k is the dimension of q .
- If $\varphi \in \{\varphi \mid \llbracket \varphi \rrbracket \in S3\}$ then q' is added by the second call to `addTargetMap`, which means that we are respecting the cubical laws $t_i(t_{k+1}(q_{k+1})) = t_k(t_i(q_{k+1}))$, for $1 \leq i \leq k$ and for some q_{n+1} for which our $q = t_i(q_{k+1})$. Then by the construction of the label it means that $\llbracket \varphi \rrbracket \in \lambda(t_{k+1}(q_{k+1}))$ which by the canonicity of these upper cells it means that $\langle \llbracket \varphi \rrbracket \rangle \in \lambda(q_{k+1})$. By axiom (A6) it means that $\llbracket \langle \varphi \rangle \rrbracket \in \lambda(q_{k+1})$ and thus, by the canonicity it means that $\langle \varphi \rangle \in \lambda(q)$ which is a contradiction with our initial assumption as the labels are atoms and hence $\neg \langle \varphi \rangle$ cannot be in the label $\lambda(q)$.
- Lastly, assume that φ is one of the formulas accumulated in the label of q' as a result of the first loop of `addTargetMap`. This means that we are respecting the cubical laws $s_i(t_{k+1}(q)) = t_k(s_i(q))$ and $\llbracket \llbracket \varphi \rrbracket \rrbracket \in \lambda(s_i(q')) = \lambda(s_i(t_{k+1}(q))) = \lambda(t_k(s_i(q)))$. By canonicity of the other cells it means that $\langle \llbracket \llbracket \varphi \rrbracket \rrbracket \rangle \in \lambda(s_i(q))$ which by axiom (A7') it means that $\llbracket \llbracket \langle \varphi \rangle \rrbracket \rrbracket \in \lambda(s_i(q))$. By canonicity again it means that $\langle \varphi \rangle \in \lambda(q)$ which is again a contradiction with our initial assumption.

Third, both q and q' are newly added by `addTargetMap`, meaning that we are looking at the second loop. The proof is the same as before as the construction of the label and axiom (A6) do all the work.

Forth, both q and q' are newly added by `addSourceMap`, which means that we are in the first loop of `addSourceMap` and there exists a q_{k+1} with $s_{k+1}(q_{k+1}) = q$ and $t_i(s_{k+1}(q_{k+1})) = q' = s_k(t_i(q_{k+1}))$ for some i . By the construction of the label of $s_{k+1}(q_{k+1})$, i.e., $\lambda(q)$, we have that for our formula $\varphi \in \lambda(q')$ there exists $\langle \varphi \rangle \in \lambda(q)$ because these are added in the label of q in the i step of the loop.

Assume that for the lifted HDA the *first canonicity condition is broken*; i.e., consider $q \in Q_n$ and assume $s_i(q) = q'$ for which $\varphi \in \lambda(q)$ and $\{\varphi\} \notin \lambda(q')$, which is the same as $\neg \{\varphi\} \in \lambda(q')$, or, by axiom (A4), $\llbracket \llbracket \neg \varphi \rrbracket \rrbracket \in \lambda(q')$. We again take cases after q .

Consider that $q \in \mathcal{H}$ and q' is added by the function `addSourceMap`. This may be done either in the first or in the second loop, but in any of the cases the construction of the labels ensures that if $\varphi \in \lambda(q)$ then $\{\varphi\} \in \lambda(q')$. The same holds for the case when both q and q' are newly added by the second call to `addSourceMap` (in the second loop).

Consider the case when both q and q' are newly added by the first call to the function `addTargetMap`. Our initial assumption says that $\llbracket \llbracket \neg \varphi \rrbracket \rrbracket \in \lambda(q')$ which means, by the iterative construction of the label of q in the loop, that $\neg \varphi \in \lambda(q)$ which is a contradiction with our initial assumption that $\varphi \in \lambda(q)$.

By now we are sure that the labeling of \mathcal{H}' is canonical. □

Lemma A.22. *A finite HDA \mathcal{H} that is potential canonical can be transformed into a canonical HDA by revealing one labeling that conforms with the potential labeling; this will be canonical. Moreover the way we generate this specific labeling does not introduce defects.*

Proof. Start with the cell that has no existential constraints, but only set constraints. These being consistent sets they can be grown to an atom. Depending on this atom build the rest of the atoms s.t. the existential constraints are respected. This can always be done.

In a finite *HDA* built using the two enrich and lift constructions starting from the minimal potential canonical *HDA* as is done in the proof of Theorem A.24 there always exists a cell with no existential constraints. Order the cells wrt. the number of existential constraints that they have. Use this order when building the labeling. \square

Lemma A.23 (repair lemma). *For any canonical \mathcal{H} that has a defect we can build a corresponding \mathcal{H}' which is canonical and does not have this defect.*

Proof. Consider that the canonical \mathcal{H} from the statement has a defect of type D1. Apply the *enriching construction* to \mathcal{H} wrt. the defective cell q_n and the formula ψ (where $\{\psi \in \lambda(q_n)\}$). The enriching lemma ensures that the new model \mathcal{H}' extends \mathcal{H} and is canonical. The enriched model \mathcal{H}' does not have the defect that \mathcal{H} had.

Consider that the canonical \mathcal{H} from the statement has a defect of type D2. Apply the *lifting construction* to \mathcal{H} wrt. the defective cell q_n (for which $\langle \psi \in \lambda(q_n) \rangle$), to obtain, cf. lifting lemma, a canonical \mathcal{H}' that extends \mathcal{H} . It is clear that the new model does not have the defect that \mathcal{H} had. \square

Theorem A.24 (completeness). *The axiomatic system of Table 2 is complete; i.e., $\forall \varphi : \models \varphi \Rightarrow \vdash \varphi$.*

Proof. Using the truth lemma A.8 for pseudo canonical and saturated *HDA*s, the proof amounts to showing that for any consistent formula φ we can build a pseudo canonical saturated \mathcal{H}_φ that has a cell labeled with an atom that contains φ . We construct \mathcal{H}_φ in steps starting with \mathcal{H}_φ^0 which contains only one cell q_0^0 of dimension 0. The construction is done in two stages: in the first stage we label the cells with constraints (i.e., we use a potential labeling); and in the second stage we explicit these constraints into corresponding atoms (i.e., we transform the potential labeling into a real labeling). The first stage builds the actual *finite HDA*, \mathcal{H}_φ , labeling it with a potential canonical labeling, striving to repair all the defects in the constraints of the cells. The final \mathcal{H}_φ is defect-free. Any finite \mathcal{H}_φ has a cell which will have no existential constraints. We start from this cell to explicit the potential labeling into atoms for each cell. During this second phase only the labels of the \mathcal{H}_φ are affected; i.e., they are transformed into atoms consistent with the potential labeling. This construction does not destroy the property of pseudo canonicity of the model \mathcal{H}_φ that we started with. Moreover, it does not introduce defects. Therefore, in the end we are left with the finite, defect-free and pseudo canonical *HDA* that we were looking for, where the label of the initial cell contains the initial formula.

Start by labeling q_0^0 with set constraints containing φ and all other formulas it implies, i.e., $\tilde{\lambda}(q_0^0) = \{\varphi\} \cup \{\psi \mid \varphi \rightarrow \psi\}$. Trivially, \mathcal{H}_φ^0 is canonical, hence also pseudo canonical and the potential labeling is potential canonical. For each defect in the potential label $\tilde{\lambda}(q_0^0)$, i.e., in the set constraints, we apply the repair lemma to obtain a new *HDA* which does not contain the repaired defect, extends the old defective *HDA*, does not introduce new defects into the just repaired potential label $\tilde{\lambda}(q_0^0)$ (it may introduce new defects in the new cells), and is pseudo canonical. The algorithm continues repairing $\tilde{\lambda}(q_0^0)$ until all defects are removed. It then continues to repair the new cells in the order that they were added, also respecting the order given below. Note that any atom that is consistent with $\{\varphi\}$ is also consistent with $\{\psi \mid \varphi \rightarrow \psi\}$.

The cells used to construct our model are picked (in the right order) from the following sets $S_i = \{q_i^j \mid j \in \omega\}$ where $i \in \omega$ corresponds to the dimension i . Any of these cells may have defects and thus, we list all the defects, i.e., all the cells, and try to repair them in increasing order (i.e., we treat first defects on level 0 and continue upwards). \square

Theorem A.25 (completeness). *The axiomatic system of Table 2 is complete. Formally $\forall \varphi : \models \varphi \Rightarrow \vdash \varphi$.*

Proof. Using the truth lemma A.4, the proof amounts to showing that for any consistent formula φ we can build a canonical saturated \mathcal{H}_φ that has a cell labeled with an atom that contains φ . We construct \mathcal{H}_φ in steps starting with \mathcal{H}_φ^0 which contains only one cell q_0^0 of dimension 0 labeled with an atom containing φ , i.e., $\lambda(q_0^0) = A_\varphi$.

Trivially, \mathcal{H}_ϕ^0 is canonical. The cells used to construct our model are picked (in the right order) from the following sets $S_i = \{q_i^j \mid j \in \omega\}$ where $i \in \omega$ corresponds to the dimension i . Any of these cells may have defects and thus, we list all the defects, i.e., all the cells, and try to repair them in increasing order (i.e., we treat first defects on level 0 and continue upwards).

At some step $n \geq 0$ in the construction we consider $\mathcal{H}_\phi^n = (Q^n, \bar{s}^n, \bar{t}^n, l^n)$ canonical. If \mathcal{H}_ϕ^n is not saturated then pick the smallest defect cell of \mathcal{H}_ϕ^n . For a D1 defect, i.e., a cell $q_k \in Q_k$ and formula $\{\psi \in \lambda(q_k)\}$, apply enrich (k, q_k, ψ) and obtain a model \mathcal{H}_ϕ^{n+1} which is canonical, cf. Lemma A.13, and does not have the D1 defect, cf. Lemma A.23. For a D2 defect apply the lifting construction to remove the defect. Moreover, any repaired defect will never appear in any extension model, independent of how many times we apply the enriching or lifting constructions. Both enriching and lifting pick their new cells from S in increasing order. We obtain \mathcal{H}_ϕ as a limit construction from all the \mathcal{H}_ϕ^n ; i.e., $\mathcal{H}_\phi = (Q, \bar{s}, \bar{t}, l)$ as $Q = \bigcup_{n \in \omega} Q^n$, $\bar{s} = \bigcup_{n \in \omega} \bar{s}^n$, $\bar{t} = \bigcup_{n \in \omega} \bar{t}^n$, $l = \bigcup_{n \in \omega} l^n$. \square